ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

«Επεξεργασία και Αναπαράσταση Βέλτιστων Διαδρομών σε Οδικά Δίκτυα»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΧΡΥΣΑΪΔΑΣ ΠΑΠΑΔΟΠΟΥΛΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

Τ.ΣΕΛΛΗΣ, Καθηγητής Ε.Μ.Π.

ΑΘΗΝΑ 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

«Επεξεργασία και Αναπαράσταση Βέλτιστων Διαδρομών σε Οδικά Δίκτυα»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

ΧΡΥΣΑΪΔΑΣ ΠΑΠΑΔΟΠΟΥΛΟΥ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

Τ.ΣΕΛΛΗΣ, Καθηγητής Ε.Μ.Π.

ΑΘΗΝΑ 2011

ΚΑΤΑΛΟΓΟΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΕΡΙΛΗΨΗ

Στόχος της παρούσας εργασίας είναι η ανάπτυξη μιας εφαρμογής που αφορά την επεξεργασία και αναπαράσταση βέλτιστων διαδρομών σε οδικά δίκτυα. Ουσιαστικά, πρόκειται για την υλοποίηση μιας client-server εφαρμογής (εφαρμογή πελάτη-εξυπηρετητή), η οποία παρέχει στο χρήστη τη δυνατότητα εύρεσης της συντομότερης διαδρομής μεταξύ του σημείου στο οποίο βρίσκεται και του σημείου στο οποίο επιθυμεί να μεταβεί.

Το πρόβλημα εύρεσης συντομότερης διαδρομής αφορά κατά κύριο λόγο, το σχεδιασμό δικτύων ή την επίλυση προβλημάτων που σχετίζονται με ήδη υπάρχοντα δίκτυα. Το ζητούμενο είναι συνήθως η έυρεση της βέλτιστης διαδρομής, μέσω της οποίας καθίσταται δυνατή η μετάβαση από ένα σημείο του δικτύου σε κάποιο άλλο, τηρουμένων κάποιων κριτηρίων τα οποία σχετίζονται με την απόσταση μεταξύ των σημείων αφετηρίας και προορισμού, με το χρόνο διάνυσης της συγκεκριμένης διαδρομής, την ελαχιστοποίηση του κόστους των διοδίων κ.λπ.

Η ραγδαία ανάπτυξη εφαρμογών πλοήγησης σε δίκτυα απαιτεί την αποτελεσματική επίλυση του προβλήματος εύρεσης συντομότερης διαδρομής, προκειμένου να εξασφαλιστεί το καλύτερο δυνατό αποτέλεσμα για τους χρήστες του εκάστοτε δικτύου. Ωστόσο, παρά τους αλγόριθμους που έχουν προταθεί για την επίλυση του προβλήματος, η αναζήτηση βέλτιστης διαδρομής σε μεγάλα και πολύπλοκα δίκτυα καθιστά τη διαδικασία επίλυσης του προβλήματος απαγορευτική και ως εκ τούτου αναζητώνται καινοτόμες και βελτιωμένες μέθοδοι επίλυσης.

Η υλοποίηση της παρούσας εφαρμογής εστιάζει, στην προσομοίωση του οδικού δικτύου της Αθήνας και τη διαγραμματική αναπαράστασή του με τη βοήθεια ενός γράφου, στην αναπαράσταση των στοιχείων του δικτύου σε μια χωρική βάση δεδομένων, στην αξιοποίηση υλοποιημένων αλγόριθμων δρομολόγησης για την εύρεση της βέλτιστης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου και στην εξαγωγή της λεκτικής και γεωμετρικής περιγραφής της σε μια ιστοσελίδα που διαθέτει διαδικτυακό χαρτογραφικό υπόβαθρο, ως αποτέλεσμα μιας διαδικασίας αναζήτησης.

Στα επιμέρους κεφάλαια του παρόντος τεύχους παρουσιάζονται ορισμένα στοιχεία από τη θεωρία των γράφων, οι κυριότεροι αλγόριθμοι αναζήτησης συντομότερης διαδρομής σε γράφους, τα βασικά χαρακτηριστικά των προγραμματιστικών εργαλείων που χρησιμοποιήθηκαν κατά την υλοποίηση της παρούσας εφαρμογής δρομολόγησης, τα επιμέρους στάδια σχεδιασμού της εφαρμογής, καθώς και κάποια συμπεράσματα που προέκυψαν μετά την ολοκλήρωση εφαρμογής.

Ευχαριστίες

Στο σημείο αυτό, θα ήθελα να ευχαριστήσω τον καθηγητή του Εθνικού Μετσοβίου Πολυτεχνείου και επιβλέποντα της παρούσας εργασίας κύριο Τ. Σελλή, για τη δυνατότητα που μου προσέφερε να ασχοληθώ με ένα ιδιαιτέρως ενδιαφέρον επιστημονικό αντικείμενο, για την εμπιστοσύνη που μου έδειξε και την κατανόηση την οποία είχα κατά τη διάρκεια εκπόνησης της παρούσας εργασίας. Ιδιαίτερες ευχαριστίες, οφείλω στον υποψήφιο διδάκτορα της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσοβίου Πολυτεχνείου κύριο Κ. Πατρούμπα, για τις πολύτιμες γνώσεις που μου προσέφερε, τις ώρες που αφιέρωσε κατά τη διάρκεια εκπόνησης της εργασίας, την αμέριστη συμπαράσταση από πλευράς του για τις όποιες δυσκολίες αντιμετώπισα κατά την πορεία ολοκλήρωσης της εργασίας και για την άψογη και άριστη συνεργασία μας.

Τέλος, θα ήθελα να ευχαριστήσω του γονείς μου και όλους τους καλούς φίλους και συναδέλφους οι οποίοι παρά τις ώρες απουσίας μου από κοντά τους λόγω των υποχρεώσεών μου, με στήριξαν και κατανόησαν τις επιθυμίες και τα όνειρά μου.

Οκτώβριος 2011

ABSTRACT

The **aim** of this master thesis is the development of an application, which concerns the processing and representation of optimum shortest paths in road networks. Substantially, the final purpose concerns the development of a client-server application which provides the user, the possibility of finding the shortest route between two points, that represent the starting and the destination point of the route respectively.

The shortest path problem concerns the process of network planning and the solution of existing problems in every kind of network. Generally, the most important subject of research is finding the shortest path which can be followed by any user of a network in the basis of some criteria such as, the geometric distance between the starting and the destination point, the time needed in order to cross the specified route, the minimization of the toll cost etc.

The rapid development of applications related to navigation services, requires the most effective solution of the shortest path problem in order to facilitate the users of a network in the best possible way. Nevertheless, in spite of the fact that many algorithms have been proposed in order the shortest path problem to be solved; the computational cost of finding shortest paths in vast and complex networks is prohibitive. Thereby, innovative and improved solutions are inquired.

The application designed in this work focuses on, the simulation of the Athens road network and its graphical representation as a graph, the representation and implementation of the network's spatial data into a spatial database, the utilization of implemented routing algorithms that find shortest paths between two or more points and its verbal and geometric representation in a web page in which a web map has been embedded.

The distinct chapters of this master thesis include: the basics of graph theory, the description of the basic shortest path algorithms, the description of the programming tools used for the development of this client-server application, the application's designing process and some conclusions arised from the whole consideration of this work.

October 2011

ΚΕΦΑΛΑΙΟ 1:ΕΙΣΑΓΩΓΗ

Το πρόβλημα εύρεσης της συντομότερης διαδρομής (shortest path problem) σε δίκτυα οδικά, σιδηροδρομικά, δίκτυα κοινής ωφέλειας, δίκτυα τηλεπικοινωνιών κ.λπ., συνιστά ένα από τα θεμελιώδη προβλήματα της σύγχρονης επιστήμης των υπολογιστών και αφορά ένα πλήθος εφαρμογών πλοήγησης και σχεδιασμού δικτύων.

Η συμβολή της θεωρίας των γράφων όσον αφορά την επίλυση προβλημάτων σε δίκτυα είναι σημαντική, καθώς η δομή ενός γράφου είναι κατάλληλη για τη μοντελοποίηση των δικτύων, των αντικειμένων τους και των συνδέσεων που υφίστανται μεταξύ των αντικειμένων αυτών.

Το πρόβλημα εύρεσης συντομότερης διαδρομής εκφράζεται κατά περίπτωση με ορισμένες διαφοροποιήσεις και ανάλογα με την περίπτωση, έχουν προταθεί διάφοροι αλγόριθμοι προσαρμοσμένοι στις ιδιαίτερες απαιτήσεις της εκάστοτε εφαρμογής. Ωστόσο, όσο μεγαλύτερο είναι ένα δίκτυο και όσο πιο πολύπλοκη καθίσταται η δομή του, ανάλογος είναι και ο βαθμός δυσκολίας που υπεισέρχεται κατά τις διαδικασίες επίλυσής του. Ως εκ τούτου, αρκετά προβλήματα που αφορούν την επίλυση δικτύων που αναπαρίστανται διαγραμματικά ως γράφοι, παραμένουν ακόμη άλυτα είτε λόγω του τεράστιου υπολογιστικού χρόνου που η επίλυσή τους προϋποθέτει είτε λόγω περιορισμών μνήμης. Σε ορισμένες περιπτώσεις, έχουν προταθεί διάφορες ad hoc προσεγγίσεις για την επίλυση αυτού του είδους των προβλημάτων, όπως για παράδειγμα το πρόβλημα του πλανόδιου πωλητή, αλλά η εξαγωγή μιας γενικής μεθοδολογίας που θα καθιστά δυνατή την επίλυση αποτελεί αντικείμενο περαιτέρω έρευνας και μελέτης.

Η ανάπτυξη εφαρμογών πλοήγησης, στις οποίες υπεισέρχεται συνήθως το ζήτημα εύρεσης συντομότερων διαδρομών σε δίκτυα, συνιστά έναν ραγδαία αναπτυσσόμενο κλάδο της πληροφορικής, με στόχο την προσφορά βελτιωμένων και αποτελεσματικών υπηρεσιών πλοήγησης στους χρήστες των εκάστοτε δικτύων. Οι διαρκώς αυξανόμενες ανάγκες των χρηστών και οι απαιτήσεις για την εξαγωγή αποτελεσμάτων ακριβείας, καθιστούν απαραίτητη τη διαρκή αναβάθμιση και βελτίωση των υπηρεσιών αυτών μέσα από την εισαγωγή καινοτόμων μεθόδων, εργαλείων και τεχνικών δρομολόγησης.

Η διαχείριση των χωρικών δεδομένων, η μοντελοποίησή τους, η χαρτογραφική αναπαράστασή τους και η επεξεργασία τους μέσα από την εφαρμογή αντίστοιχων αλγόριθμων δρομολόγησης συνιστούν μερικές από τις βασικότερες παραμέτρους που υπεισέρχονται στις διαδικασίες σχεδιασμού εφαρμογών δρομολόγησης. Η αποτελεσματική διαχείριση και ενσωμάτωση των παραπάνω παραμέτρων σε ολοκληρωμένες εφαρμογές δρομολόγησης απαιτεί συνήθως τη συνδυασμένη χρήση υπολογιστικών εργαλείων, όπως τα Συστήματα Διαχείρισης Χωρικών Βάσεων Δεδομένων, οι διαδικτυακές χαρτογραφικές υπηρεσίες κ.λπ., προκειμένου να καταστεί δυνατή η πολυδιάστατη διαχείριση των χωρικών δεδομένων που αφορούν τα στοιχεία και τα χαρακτηριστικά του εκάστοτε χωρικού δικτύου.

Οι εφαρμογές πλοήγησης προσφέρουν στους χρήστες τη δυνατότητα εύρεσης της βέλτιστης διαδρομής μέσω της οποίας είναι δυνατή η μετάβασή τους από το σημείο που βρίσκονται σε κάποιο άλλο σημείο προορισμού, της χρονικής διάρκειας που απαιτεί η διάσχιση μιας συγκεκριμένης διαδρομής, της απόστασης που υφίσταται μεταξύ δύο σημείων του δικτύου κ.ά. Συνιστούν τεχνολογίες αιχμής οι οποίες βελτιώνονται διαρκώς προκειμένου να ανταποκρίνονται πλήρως στις αυξανόμενες απαιτήσεις των χρηστών, μέσα από το σχεδιασμό νέων βελτιωμένων αλγόριθμων και την εισαγωγή καινοτόμων μεθόδων και τεχνικών διαχείρισης χωρικών δεδομένων και χωρικών δικτύων.

1.1 Στόχος και Αντικείμενο Εργασίας

Η παρούσα εργασία εστιάζει στην επεξεργασία και αναπαράσταση βέλτιστων διαδρομών σε οδικά δίκτυα. Αφορά στο σχεδιασμό μιας ολοκληρωμένης client-server εφαρμογής, μέσω της οποίας ένας χρήστης του οδικού δικτύου της Αθήνας έχει τη δυνατότητα αναζήτησης της συντομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου. Ιδιαίτερη έμφαση δίνεται στην επεξεργασία συναρτήσεων που υλοποιούν αλγόριθμους δρομολόγησης σε δίκτυα, η εφαρμογή των οποίων έχει ως αποτέλεσμα την εύρεση της βέλτιστης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου.

Το πρώτο στάδιο υλοποίησης της εφαρμογής αφορά την αναπαράσταση του βασικού οδικού δικτύου της Αθήνας σε μια χωρική βάση δεδομένων και την επεξεργασία συναρτήσεων που υλοποιούν τρεις αλγόριθμους δρομολόγησης, τον αλγόριθμο του Dijkstra, τον αλγόριθμο A* και τον αλγόριθμο Traveling Sales Person. Πρόκειται για

τρεις αλγόριθμους αναζήτησης συντομότερης διαδρομής σε δίκτυα, κάθε ένας από τους οποίους ενσωματώνει ορισμένα ιδιαίτερα χαρακτηριστικά που τον διαφοροποιούν από τους υπόλοιπους και εφαρμόζεται κατά περίπτωση για την αναζήτηση της βέλτιστης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων ενός δικτύου που αναπαρίσταται ως γράφος.

Το δεύτερο στάδιο της εφαρμογής περιλαμβάνει το σχεδιασμό μιας ιστοσελίδας, η οποία επιτρέπει στο χρήστη την αναζήτηση βέλτιστων διαδρομών μεταξύ σημείων που αυτός επιλέγει σε ένα διαδικτυακό χάρτη. Μέσω της ιστοσελίδας, ο χρήστης έχει τη δυνατότητα επιλογής των σημείων που ορίζουν την προς αναζήτηση διαδρομή σε χαρτογραφικό υπόβαθρο όπου απεικονίζεται το οδικό δίκτυο της Αθήνας. Παράλληλα, έχει τη δυνατότητα επιλογής του αλγόριθμου βάσει του οποίου πρόκειται να υπολογιστεί η συντομότερη διαδρομή μεταξύ των σημείων που κάθε φορά επιλέγει, της παραμέτρου βάσει της οποίας πρόκειται να υπολογιστεί το κόστος διάσχισης της ζητούμενης διαδρομής (χρόνος ή γεωμετρική απόσταση) καθώς και το τμήμα του οδικού δικτύου που πρόκειται να συμπεριληφθεί στους υπολογισμούς με κριτήριο τη χρήση ή όχι των οδικών τμημάτων που βρίσκονται στην περιοχή του δακτυλίου. Το σύνολο των απαιτούμενων υπολογισμών υλοποιείται στη βάση δεδομένων, με την οποία συνδέεται η διαδικτυακή εφαρμογή, μέσω μιας δεύτερης ενδιάμεσης εφαρμογής που τρέχει στον *server* και μεταφέρει δεδομένα από την ιστοσελίδα στη βάση και αντίστροφα.

Τα αποτελέσματα κάθε αναζήτησης, αναπαρίστανται στην ιστοσελίδα υπό μορφή γεωμετρικής και λεκτικής περιγραφής. Πιο συγκεκριμένα, η ζητούμενη διαδρομή αναπαρίσταται διανυσματικά ως ένα *kml* αρχείο επάνω στο διαδικτυακό χάρτη που περιλαμβάνει η ιστοσελίδα και λεκτικά, σε ένα δυναμικό πίνακα ο οποίος περιλαμβάνει τις ονομασίες των επιμέρους γεωμετρικών συνδέσμων που πρέπει να διασχίσει ο χρήστης προκειμένου να μεταβεί από το σημείο αφετηρίας στο σημείο προορισμού καθώς και το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου.

Τέλος, η προτεινόμενη λύση αξιολογείται ως προς την αποτελεσματικότητά της, τα πλεονεκτήματα και τα μειονεκτήματά της, σε σύγκριση με μια σειρά λογισμικών που χρησιμοποιούνται για το σχεδιασμό και την υλοποίηση εφαρμογών δρομολόγησης. Παράλληλα, εξετάζεται η δυνατότητα εισαγωγής πρόσθετων δυνατοτήτων ούτως ώστε η εφαρμογή να καταστεί πιο ελκυστική.

1.2 Διάρθρωση Εργασίας

Στόχος της εργασίας είναι ο σχεδιασμός μιας εφαρμογής δρομολόγησης, η οποία καθιστά εφικτή την επεξεργασία και αναπαράσταση βέλτιστων διαδρομών στο οδικό δίκτυο της Αθήνας. Στα επιμέρους κεφάλαια της εργασίας, αναπτύσσεται το θεωρητικό υπόβαθρο που πλαισιώνει τους γράφους και τους αλγόριθμους που έχουν προταθεί για την επίλυση προβλημάτων σε γράφους και παρουσιάζονται αναλυτικά τα στάδια ανάπτυξης της εφαρμογής δρομολόγησης που σχεδιάστηκε. Παράλληλα, παρατίθενται τα συμπεράσματα που προκύπτουν από τη συνολική θεώρηση της εφαρμογής ενώ στο Παράρτημα, παρουσιάζονται οι κώδικες που αναπτύχθηκαν κατά τη διαδικασία σχεδιασμού της εφαρμογής.

Αναλυτικότερα:

Στο **δεύτερο κεφάλαιο**, παρουσιάζεται το εννοιολογικό πλαίσιο και οι βασικοί ορισμοί που αφορούν στη θεωρία των γράφων. Περιγράφονται οι κυριότερες ιδιότητες και τα χαρακτηριστικά των γράφων, οι επιμέρους κατηγορίες στις οποίες ταξινομούνται και οι δομές δεδομένων που χρησιμοποιούνται για την αναπαράστασή τους στον Η/Υ. Παράλληλα, παρουσιάζονται ορισμένα βασικά προβλήματα η επίλυση των οποίων ανάγεται στην επίλυση προβλήματος σε γράφο και οι κυριότεροι αλγόριθμοι που έχουν προταθεί για την επίλυση προβλημάτων σε γράφους.

Το **τρίτο κεφάλαιο** εστιάζει στα σημαντικότερα χαρακτηριστικά των Συστημάτων Διαχείρισης Χωρικών Βάσεων Δεδομένων, στη σχέση τους με τα Συστήματα Γεωγραφικών Πληροφοριών, στις διαφοροποιήσεις που παρουσιάζουν σε σχέση με τα παραδοσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων και στις επιπλέον δυνατότητες που παρέχουν στους χρήστες.

Στο **τέταρτο κεφάλαιο**, περιγράφονται συνοπτικά τα χαρακτηριστικά των λογισμικών ανοικτού κώδικα PostgreSQL/ PostGIS + pgRouting, τα οποία χρησιμοποιήθηκαν για την υλοποίηση της βάσης δεδομένων που δημιουργήθηκε στα πλαίσια της παρούσας εφαρμογής. Παράλληλα, παρουσιάζονται τα στάδια υλοποίησης της βάσης και οι πρόσθετες διαδικασίες που έλαβαν χώρα στο περιβάλλον τις PostgreSQL προκειμένου να ανταποκρίνεται πλήρως στις απαιτήσεις της σχεδιαζόμενης εφαρμογής.

Στο **πέμπτο κεφάλαιο**, παρουσιάζονται οι διαδικασίες υλοποίησης της διαδικτυακής εφαρμογής. Περιγράφονται τα επιμέρους στάδια ανάπτυξης της εφαμογής και τα

προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν για το σκοπό αυτό. Επιπρόσθετα, παρουσιάζεται η ανάπτυξη μιας εφαρμογής που καθιστά εφικτή την ανταλλαγή δεδομένων ανάμεσα στον *server*, τη βάση δεδομένων και την ιστοσελίδα. Πρόκειται για την ανάπτυξη ενός PHP κώδικα, ο οποίος αποτελεί το «συνδετικό κρίκο» μεταξύ της βάσης δεδομένων όπου υλοποιείται η επεξεργασία των δεδομένων και της ιστοσελίδας που συνιστά το περιβάλλον δραστηριότητας του χρήστη.

Το **έκτο κεφάλαιο** εστιάζει στην παρουσίαση των συμπερασμάτων που προέκυψαν μετά την ολοκλήρωση του σχεδιασμού της εφαρμογής και τη συνολική θεώρηση των διαδικασιών που εφαρμόστηκαν κατά τη διαδικασία σχεδιασμού της.

Τέλος, ακολουθεί η παράθεση της βιβλιογραφίας που χρησιμοποιήθηκε για την άντληση της απαραίτητης για την ολοκλήρωση της εφαρμογής πληροφορίας, το γλωσσάριο επεξήγησης των αγγλικών όρων που χρησιμοποιούνται στο κείμενο και το παράρτημα που περιλαμβάνει το σύνολο των κωδίκων που υλοποιήθηκαν στα πλαίσια σχεδιασμού της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2: ΣΤΟΙΧΕΙΑ ΑΠΟ ΤΗ ΘΕΩΡΙΑ ΓΡΑΦΩΝ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ ΓΡΑΦΩΝ

Το κεφάλαιο αυτό εστιάζει στην παρουσίαση των θεμελιωδών ορισμών, εννοιών και κανόνων που αφορούν στη θεωρία των γράφων, όπως αυτή καθιερώθηκε μέσα από την επιστήμη των μαθηματικών και επεκτάθηκε σε εφαρμογές πληροφορικής για τη μοντελοποίηση πάσης φύσεως δικτύων. Επιπλέον, περιλαμβάνει την περιγραφή βασικών αλγόριθμων που έχουν αναπτυχθεί για την επίλυση γεωγραφικών κυρίως προβλημάτων, η αναπαράσταση των οποίων υλοποιείται με τη βοήθεια γράφων. Ο σχεδιασμός αλγόριθμων που επιλύουν προβλήματα αυτής της κατηγορίας εστιάζει, αφενός μεν στη διαδοχική αναζήτηση εναλλακτικών λύσεων που ανταποκρίνονται στις απαιτήσεις του εκάστοτε προβλήματος, αφετέρου δε στην αξιολόγηση των λύσεων και την τελική επιλογή της βέλτιστης η οποία ικανοποιεί στο μέγιστο βαθμό τα κριτήρια που έχουν τεθεί.

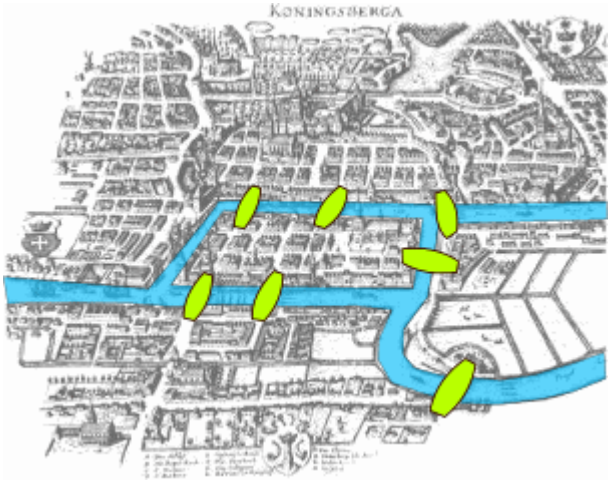
Αρχικά, παρουσιάζεται το εννοιολογικό πλαίσιο σχετικά με τον ακριβή και επιστημονικό ορισμό του γράφου και των δομικών του στοιχείων. Στη συνέχεια, ακολουθεί η παρουσίαση των βασικών κατηγοριών ταξινόμησης των γράφων ανάλογα με τα χαρακτηριστικά τους και η περιγραφή των βασικών ιδιοτήτων τους. Παράλληλα, αναπτύσσονται ζητήματα σχετικά με τις δομές δεδομένων η εφαρμογή των οποίων καθιστά δυνατή την αναπαράσταση ενός γράφου σε Η/Υ και εξετάζεται το πρόβλημα της μοντελοποίησης δικτύων με τη βοήθεια γράφων μέσα από την παρουσίαση σχετικών παραδειγμάτων. Ακολουθεί η παρουσίαση ορισμένων θεμελιωδών προβλημάτων όπως το πρόβλημα διάσχισης γράφου (graph traversal), το πρόβλημα εύρεσης συντομότερης διαδρομής (shortest path problem), το πρόβλημα εύρεσης της μεταβατικής κλειστότητας γράφου (transitive closure), το πρόβλημα του πλανόδιου πωλητή (travelling salesman problem) και το πρόβλημα χρωματισμού πολυγώνων σε θεματικούς χάρτες (coloured polygons in thematic maps). Αναλύονται τα δεδομένα και περιγράφεται ο προβληματισμός που αναπτύσσεται κατά περίπτωση. Στη συνέχεια, παρουσιάζονται οι κυριότεροι αλγόριθμοι που έχουν προταθεί για την επίλυση προβλημάτων γεωγραφικής φύσεως, όπως τα παραπάνω, οι βασικές λειτουργίες και η βασική δομή κάθε αλγόριθμου.

Παράλληλα, για ορισμένους από αυτούς δίνεται ο ψευδοκώδικας, όπου παρουσιάζονται συνοπτικά και ημι- προγραμματιστικά τα επιμέρους βήματα και οι διαδικασίες που περιλαμβάνονται σε κάθε αλγόριθμο.

Στόχος του εισαγωγικού αυτού κεφαλαίου είναι η κατανόηση και αποσαφήνιση του θεωρητικού υποβάθρου που πλαισιώνει τη θεωρία γράφων, στην οποία στηρίζεται κατά κύριο λόγο η υλοποίηση της εφαρμογής που εκπονήθηκε στα πλαίσια της παρούσας εργασίας.

2.1 Βασικοι Ορισμοί – Εννοιολογικό Πλαίσιο

Οι βάσεις της θεωρίας των γράφων τέθηκαν από την επιστήμη των διακριτών μαθηματικών στο πρώτο μισό περίπου του $18^{ου}$ αιώνα. Η διαχρονική εξέλιξη και ανάπτυξη του συγκεκριμένου πεδίου των μαθηματικών κατέστησε το γράφο μια δομή, η οποία υιοθετείται πλέον από διαφορετικούς επιστημονικούς κλάδους καθώς είναι κατάλληλη για την αναπαράσταση πάσης φύσεως δικτύων και συστημάτων τα στοιχεία των οποίων συνδέονται μεταξύ τους με σχέσεις αλληλεπίδρασης. Η πρώτη ιστορικά δημοσιευμένη, επιστημονική εργασία που σήμανε την απαραχή της μελέτης των γράφων και των προβλημάτων η αναπαράσταση των οποίων υλοποιείται με τη βοήθεια γράφων, ήταν αυτή που δημοσιεύτηκε το 1736 από τον Leonhard Euler και αφορούσε ένα χωρικό πρόβλημα που εντοπιζόταν στην πόλη Koenigsberg της Πρωσίας (σημερινή επαρχία Kalinigrad στη Ρωσία). Η πόλη του Koenigsberg περιστοιχιζόταν από ποτάμια ύδατα και συνδεόταν με την υπόλοιπη ενδοχώρα μέσω επτά γεφυρών. Το πρόβλημα τέθηκε ως εξής: «Είναι δυνατό κάποιος να διασχίσει και τις επτά γέφυρες της πόλης επιστρέφοντας στο ίδιο ακριβώς σημείο από το οποίο ξεκίνησε τη διαδρομή του περνώντας από κάθε γέφυρα το πολύ μια φορά;».



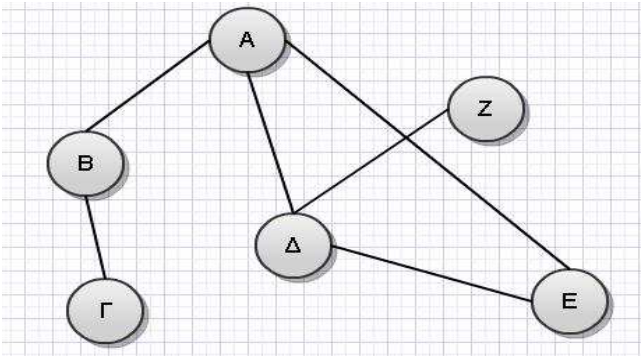
Διάγραμμα 2-1: Οι Επτά Γέφυρες του Koenigsberg
Πηγή:[URL7]

Η διατύπωση του προβλήματος παραπέμπει σε παρόμοια προβλήματα που τίθενται σήμερα και αφορούν δίκτυα και την επίλυση προβλημάτων που εντοπίζονται σε αυτά, όπως για παράδειγμα η εύρεση της βέλτιστης διαδρομής μεταξύ δύο σημείων ενός δικτύου, το πρόβλημα του πλανόδιου πωλητή, η αναζήτηση πιθανών διαδρομών που συνδέουν μεταξύ τους δύο ή περισσότερα σημεία κ.ά. Σε αντίθεση με τη μελέτη μεμονωμένων χωρικών αντικειμένων, όπου οι σχέσεις μεταξύ τους αναλύονται με βασικό άξονα την εγγύτητα, η μελέτη χωρικών αντικειμένων που συναποτελούν ένα δίκτυο βασίζεται στις σχέσεις συνδεσιμότητας που αναπτύσσονται μεταξύ των επιμέρους στοιχείων του δικτύου.

Το αντικείμενο της θεωρίας γράφων είναι η μελέτη των γράφων, μαθηματικών δομών δηλαδή, που χρησιμοποιούνται για τη μοντελοποίηση και την αναπαράσταση των σχέσεων που υφίστανται μεταξύ ζευγών αντικειμένων. Σύμφωνα με τον ορισμό που προέρχεται από τα μαθηματικά, ένας γράφος $G = (V, E)$ συνιστά μια αναπαράσταση ενός συνόλου αντικειμένων που συνδέονται μεταξύ τους. Τα αντικείμενα καλούνται κόμβοι ή κορυφές (vertices) του γράφου και συμβολίζονται με V και οι συνδέσεις που υφίστανται ανάμεσα σε ένα ζεύγος κόμβων ονομάζονται ακμές (edges) του γράφου και συμβολίζοναι με E. Οι Aldous and Wilson ορίζουν το γράφο ως ένα διάγραμμα αποτελούμενο από σημεία που ονομάζονται κόμβοι, τα οποία συνδέονται μεταξύ τους με γραμμές που ονομάζονται ακμές, ενώ κάθε ακμή συνδέει ακριβώς δύο κόμβους [1].

Κάθε ακμή του γράφου ορίζεται από ένα ζεύγος κόμβων που κείνται στα άκρα της και οι οποίοι καλούνται άκρα της ακμής. Κάθε κόμβος του γράφου χαρακτηρίζεται από το βαθμό του, ο οποίος ορίζεται ίσος με τον αριθμό των ακμών που συνδέονται με το

συγκεκριμένο κόμβο. Δύο ακμές που συνδέονται στον ίδιο κόμβο καλούνται γειτονικές ακμές, ενώ δύο κόμβοι που συνδέονται μεταξύ τους μέσω μιας κοινής ακμής καλούνται γειτονικοί κόμβοι αντίστοιχα. Το σύνολο των ακμών ενός γράφου δύναται να είναι κενό ενώ το σύνολο των κόμβων είναι απαραιτήτως μη- κενό. Οι ακμές ενός γράφου μπορεί να είναι κυκλικές (loops), δηλαδή να καταλήγουν στον ίδιο κόμβο από τον οποίο ξεκινούν. Στην περίπτωση που μια ακμή συνδέεται με έναν κόμβο, ο οποίος με τη σειρά του δε συνδέεται μέσω ακμής με κάποιο άλλο κόμβο του γράφου, καλείται τυφλή ακμή, ενώ μια ακμή, η οποία δε συνδέεται με άλλη ακμή και τα άκρα της είναι βαθμού ένα, καλείται αιωρούμενη ακμή. Ακμές που συνδέουν ακριβώς το ίδιο ζεύγος κόμβων καλούνται πολλαπλές ακμές. Τέλος, θα πρέπει να αναφερθεί ότι ένας κόμβος δύναται να ανήκει σε ένα γράφο, χωρίς απαραίτητα να ανήκει ταυτόχρονα σε κάποια από τις ακμές του γράφου [1], [28].



Διάγραμμα 2-2: Σχηματική Αναπαράσταση Γράφου
Πηγή: [URL5]

2.2 Τύποι Γράφων

Οι γράφοι εμφανίζονται διαφοροποιημένοι μεταξύ τους, ανάλογα με την ιδιαίτερη δομή και τα χαρακτηριστικά τους. Ως εκ τούτου, ταξινομούνται σε επιμέρους κατηγορίες στη βάση ενός αριθμού κριτηρίων που τίθενται κατά περίπτωση, όπως για παράδειγμα η ανάθεση ή μη φοράς στις ακμές του γράφου, η συνδεσιμότητα του γράφου, η ανάθεση βαρών στις ακμές του γράφου κ.λπ.

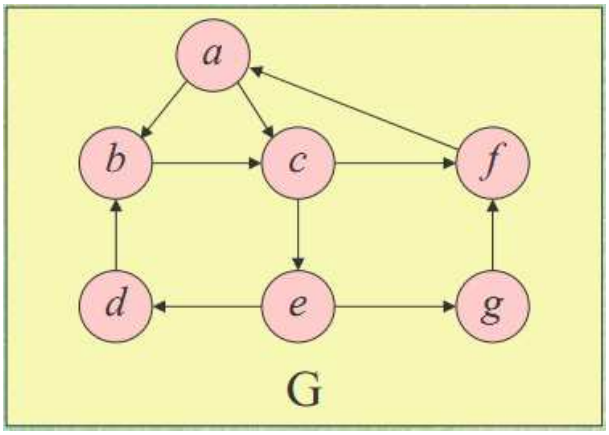
Ανάλογα με τη δομή και τα χαρακτηριστικά τους, οι γράφοι υιοθετούνται για τη μοντελοποίηση διαφορετικής φύσεως προβλημάτων. Ένας κατευθυνόμενος γράφος για

παράδειγμα, συνιστά την καταλληλότερη δομή για τη μοντελοποίηση δικτύων ενώ ένας απλός μη κατευθυνόμενος γράφος, αποτελεί κατάλληλη δομή για την αναπαράσταση των χημικών δεσμών που υφίστανται μεταξύ των ατόμων ενός μορίου. Κάθε κατηγορία γράφων χαρακτηρίζεται από την ιδιαίτερη διαγραμματική αναπαράσταση των γράφων που ανήκουν σ'αυτή, τα ιδιαίτερα χαρακτηριστικά και τις ιδιότητες των δομικών στοιχείων των γράφων και τους κανόνες που τίθενται από τα μαθηματικά και χαρακτηρίζουν τους γράφους κάθε κατηγορίας.

Στη συνέχεια, παρουσιάζονται οι βασικότερες κατηγορίες στις οποίες ταξινομούνται οι γράφοι ανάλογα με τη δομή και τα χαρακτηριστικά τους.

- **Απλοί Γράφοι (Simple Graphs)**: Ένας απλός γράφος $G = (V, E)$ ορίζεται ως ένα διάγραμμα αποτελούμενο από κόμβους και ακμές, ενώ μεταξύ δύο κόμβων του γράφου υφίσταται μία και μόνο μία ακμή. Ένας απλός γράφος δεν περιλαμβάνει κυκλικές ακμές [1].

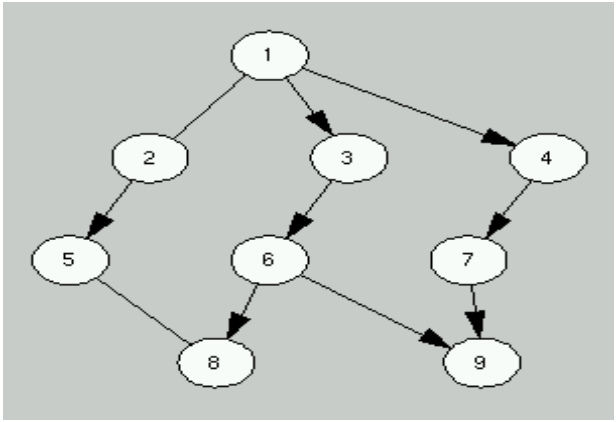
- **Κατευθυνόμενοι Γράφοι (Directed Graphs)**: Ένας γράφος ορίζεται ως κατευθυνόμενος γράφος $G = (V, A)$ όταν οι ακμές που συνδέουν τους κόμβους του είναι προσανατολισμένες προς μια κατεύθυνση (φορά), οπότε και αυτές με τη σειρά τους χαρακτηρίζονται ως κατευθυνόμενες ακμές (directed edges) ή τόξα (arcs). Ένα τόξο $a = (x, y)$ που ανήκει σε έναν κατευθυνόμενο γράφο, έχει κατεύθυνση από τον κόμβο x προς τον κόμβο y. Ο κόμβος y καλείται άμεσος διάδοχος (direct successor) του x και ο κόμβος x άμεσος προκάτοχος (direct predecessor) του κόμβου y [28].



Διάγραμμα 2-3: Κατευθυνόμενος Γράφος

Πηγή: [URL20]

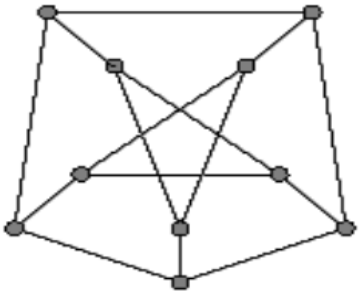
- **Μεικτοί Γράφοι (Mixed Graphs)**: Μεικτός καλείται ένας γράφος *G = (V, E, A)* ο οποίος είναι δυνατό να περιλαμβάνει ταυτόχρονα κατευθυνόμενες και μη κατευθυνόμενες ακμές. Οι γράφοι αυτού του είδους συνιστούν ειδική περίπτωση γράφου [URL16].



Διάγραμμα 2-4: Μεικτός Γράφος
Πηγή: [URL15]

- **Συνδεδεμένοι Γράφοι (Connected Graphs) και μη Συνδεδεμένοι Γράφοι**: Ως συνδεδεμένος χαρακτηρίζεται ένας γράφος, στον οποίο υπάρχουν ένα ή περισσότερα μονοπάτια μέσω των οποίων συνδέονται δύο οποιοιδήποτε κόμβοι του γράφου. Ως μη συνδεδεμένος χαρακτηρίζεται ένας γράφος, στον οποίο δεν υφίσταται απαραίτητα σύνδεση μεταξύ του συνόλου των κόμβων που περιλαμβάνονται στο γράφο. Σε ένα μη συνδεδεμένο γράφο, δύο κόμβοι *u* και *v* ονομάζονται συνδεδεμένοι εάν υφίσταται στο γράφο μονοπάτι μέσω του οποίου συνδέονται οι δύο κόμβοι. Η αφαίρεση κόμβων ή ακμών δύναται να καταστήσει ένα συνδεδεμένο γράφο, μη συνδεδεμένο. Ένας γράφος είναι ισχυρά συνδεδεμένος (strongly connected graph) όταν, δύο οποιοιδήποτε κόμβοι του συνδέονται μέσω ενός μονοπατιού είτε αυτό κατευθύνεται από τον κόμβο *u* στον κόμβο *v* είτε έχει αντίστροφη κατεύθυνση από τον *v* στον *u*. Στην περίπτωση που οι κατευθυνόμενες ακμές ενός μη συνδεδεμένου γράφου αντικατασταθούν με μη κατευθυνόμενες ακμές, ένας γράφος που πριν την αντικατάσταση ήταν μη συνδεδεμένος δύναται να καταστεί συνδεδεμένος αλλά όχι ισχυρά συνδεδεμένος γράφος [1], [URL17], [URL15].

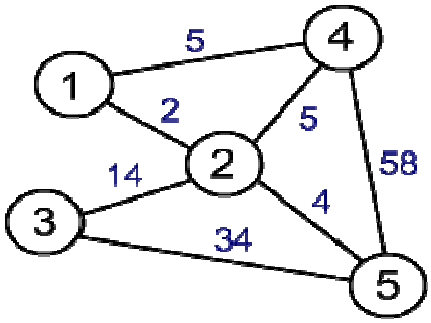
- **Κανονικοί Γράφοι (Regular Graphs)**: Ένας γράφος ορίζεται ως κανονικός όταν όλοι οι κόμβοι του έχουν ίσο αριθμό γειτονικών κόμβων. Στην περίπτωση αυτή, όλοι οι κόμβοι του γράφου έχουν ακριβώς τον ίδιο βαθμό [1].



Διάγραμμα 2-5: Κανονικός Γράφος
Πηγή: [URL19]

- **Πλήρεις Γράφοι (Complete Graphs)**: Καλούνται οι γράφοι όπου μεταξύ κάθε ζεύγους κόμβων υφίσταται οπωσδήποτε μια σύνδεση. Οι γράφοι αυτής της κατηγορίας, περιλαμβάνουν το μεγιστο δυνατό αριθμό ακμών [URL17].

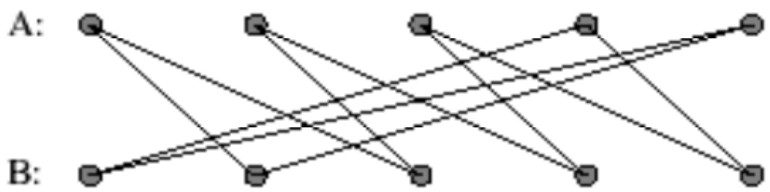
- **Σταθμισμένοι Γράφοι (Weighted Graphs)**: Ένας σταθμισμένος γράφος, είναι ένας γράφος στις ακμές του οποίου ανατίθενται βάρη. Τα βάρη αυτά μπορεί να αναπαριστούν το κόστος μιας διαδρομής, το μήκος της, το χρόνο που απαιτείται για τη συνολική διάσχιση της συγκεκριμένης διαδρομής κ.λπ. [1], [28].



Διάγραμμα 2-6: Σταθμισμένος Γράφος
Πηγή: [URL1]

- **Επίπεδοι (Planar Graphs) και μη Επίπεδοι Γράφοι**. Ένας γράφος καλείται επίπεδος όταν μπορεί να σχεδιαστεί στο επίπεδο χωρίς να διασταυρώνονται οι πλευρές του. Δύο οποιεσδήποτε ακμές ενός επίπεδου γράφου συναντώνται μόνο σε προσκείμενους ή τερματικούς κόμβους. Στην περίπτωση που οι ακμές και οι κόμβοι του γράφου κείνται στο χώρο, ο γράφος καλείται μη επίπεδος. Στις τομές των ακμών ενός μη επίπεδου γράφου δεν παρεμβάλλεται κόμβος [1], [URL19].

- **Διμερείς Γράφοι (Bipartite Graphs)**: Ένας γράφος καλείται διμερής όταν οι κορυφές του μπορούν να διαιρεθούν σε δύο σύνολα έτσι ώστε κάθε στοιχείο του ενός να συνδέεται με κάποιο στοιχείο του άλλου. Δύο στοιχεία που ανήκουν στο ίδιο σύνολο δε συνδέονται μεταξύ τους [URL19].



Διάγραμμα 2-7: Διμερής Γράφος
Πηγή: [URL19]

- **Κυκλικοί Γράφοι (Cycle Graphs)**: Ένας γράφος καλείται κυκλικός όταν αποτελείται από έναν κύκλο, δηλαδή έναν κόμβο και μια κυκλική ακμή όπου η αρχή και το πέρας της είναι ο μοναδικός κόμβος του γράφου. Επίσης, ένας κυκλικός κόμβος ορίζεται και ως μια κλειστή «αλυσίδα» ακμών που συνδέονται μεταξύ τους, ενώ ο κόμβος αφετηρίας της πρώτης ακμής είναι ο ίδιος με τον κόμβο πέρατος της τελευταίας ακμής [URL17].

2.3 Ιδιότητες Γράφων

Το παρόν εδάφιο αφορά σε μια συνοπτική παρουσίαση των κυριότερων ιδιοτήτων των γράφων και των δομικών τους στοιχείων. Οι ιδιότητες των γράφων απορρέουν κατά

βάση από τη δομή και τα ιδιαίτερα χαρακτηριστικά κάθε κατηγορίας στην οποία εντάσσονται. Κάθε κατηγορία γράφων χαρακτηρίζεται από ένα πλήθος ιδιοτήτων που αφορούν αποκλειστικά τους γράφους της εκάστοτε κατηγορίας, ενώ ορισμένες βασικές ιδιότητες αφορούν το σύνολο των γράφων ανεξαρτήτως κατηγορίας.

Οι βασικές ιδιότητες που χαρακτηρίζουν κάθε γράφο είναι αυτές που σχετίζονται με το βαθμό του γράφου και των δομικών του στοιχείων, το μέγεθος του γράφου και τις σχέσεις γειτνίασης που αναπτύσσονται μεταξύ των κόμβων και των ακμών του.

Ο βαθμός ενός κόμβου ορίζεται ίσος με το πλήθος των ακμών που συνδέονται με αυτόν τον κόμβο, ενώ ο βαθμός ενός γράφου αντίστοιχα ορίζεται ίσος με το συνολικό αριθμό των κόμβων που περιλαμβάνει και συμβολίζεται με $|V|$. Το μέγεθος ενός γράφου ορίζεται ίσο με τον αριθμό των ακμών του και συμβολίζεται με $|E|$. Ο βαθμός ενός κόμβου που είναι το πέρας μιας τυφλής ακμής είναι πάντα ίσος με ένα. Ίσος με ένα είναι επίσης και ο βαθμός των δύο κόμβων που ορίζουν μια αιωρούμενη ακμή. Σε κάθε γράφο, το άθροισμα των βαθμών των κόμβων του ισούται με το διπλάσιο του αριθμού των ακμών του [1].

Οι ιδιότητες που αφορούν σχέσεις γειτνίασης σε γράφους, σχετίζονται με τις σχέσεις γειτνίασης που αναπτύσσονται μεταξύ των δομικών στοιχείων των γράφων. Έτσι, δύο ακμές που συνδέονται με τον ίδιο κόμβο καλούνται γειτονικές ακμές (adjacent edges), ενώ δύο κόμβοι που συνδέονται με μια κοινή ακμή καλούνται γειτονικοί κόμβοι (adjacent nodes). Μία ακολουθία γειτονικών ακμών σε ένα γράφο, κάθε μια από τις οποίες έχει έναν κοινό κόμβο με την ακμή που προηγείται, ορίζει ένα μονοπάτι (path) που συνδέει δύο οποιουσδήποτε κόμβους του γράφου [20].

Κάθε ακμή που ανήκει σε ένα σταθμισμένο γράφο χαρακτηρίζεται από το βάρος που της αντιστοιχεί και το οποίο ανάλογα με την περίπτωση μπορεί να συμβολίζει το κόστος διάσχισής της, το μήκος της ή οποιοδήποτε άλλο μέγεθος απαιτεί η εκάστοτε εφαρμογή. Το συνολικό βάρος ενός σταθμισμένου γράφου ισούται με το άθροισμα των βαρών του συνόλου των ακμών του γράφου [1].

Επιπρόσθετα, από έναν ή περισσότερους υπάρχοντες γράφους δύναται να προκύψει ένας νέος γράφος μέσα από την πρόσθεση ή την αφαίρεση κόμβων ή ακμών, τη συγχώνευση κόμβων ή τη σύνδεση κόμβων που ανήκουν σε διαφορετικούς γράφους. Οι διαδικασίες δημιουργίας ενός νέου γράφου από έναν ή περισσότερους υφιστάμενους γράφους συνιστούν στοιχειώδεις λειτουργίες οι βασικότερες των οποίων κατηγοριοποιούνται ως ακολούθως:

- Στοιχειώδεις (elementary) λειτουργίες: Αφορούν τη δημιουργία ενός νέου γράφου από έναν ήδη υπάρχοντα γράφο μέσω μιας απλής τοπικής αλλαγής όπως η πρόσθεση ή διαγραφή κόμβου ή ακμής, η συγχώνευση κόμβων κ.λπ.
- «Μοναδιαίες» (unary) λειτουργίες: Αφορούν τη δημιουργία ενός νέου γράφου, γραμμικού, συμπληρωματικού κ.α. από ήδη υπάρχοντα γράφο.
- Δυαδικές (binary) λειτουργίες: Αφορούν τη δημιουργία ενός νέου γράφου από δύο προϋπάρχοντες αρχικούς γράφους.

Τέλος, έχει αποδειχθεί ότι για έναν επίπεδο γράφο, το σύνολο των πιθανών αναπαραστάσεών του απαρτίζεται από το ίδιο πλήθος πολυγώνων. Συνεπώς, εάν ένας επίπεδος γράφος αποτελείται από κ κόμβους, α ακμές και π πολύγωνα ισχύει η σχέση:

$$\pi - \alpha + \kappa = 2$$

Η παραπάνω σχέση είναι γνωστή ως κριτήριο του Euler και αποτελεί έναν απλό έλεγχο για τη μη-επιπεδότητα ενός γράφου. Συνιστά ικανή αλλά όχι αναγκαία συνθήκη για την επιπεδότητα, δηλαδή κάποιος γράφος που δεν ικανοποιεί το κριτήριο είναι γράφος μη-επίπεδος. Ωστόσο, ένας γράφος που ικανοποιεί το κριτήριο του Euler δεν είναι απαραίτητα επίπεδος. Η γενικευμένη μορφή του κριτηρίου του Euler, προκειμένου να ισχύει και για μη-συνδεδεμένους γράφους είναι η ακόλουθη:

$$\pi - \alpha + \kappa - \mu = 1$$

όπου μ, ο αριθμός των συστατικών μερών που απαρτίζουν ένα μη-συνδεδμένο γράφο [28].

2.4 Δομές Δεδομένων για την Αναπαράσταση Γράφων

Η αναπαράσταση των γράφων στον υπολογιστή απαιτεί την εφαρμογή ειδικών δομών δεδεομένων, κάθε μια από τις οποίες χαρακτηρίζεται από τα πλεονεκτήματα και τα μειονεκτήματά της. Όπως αναφέρεται και στη συνέχεια του παρόντος εδαφίου, ανάλογα με το πρόβλημα που τίθεται κάθε φορά προς επίλυση, εφαρμόζεται η αντίστοιχη δομή

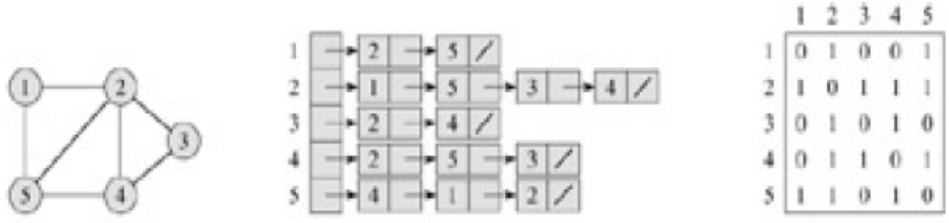
δεδομένων που εξασφαλίζει είτε την ταχύτερη επίλυση του προβλήματος είτε την εξοικονόμηση χώρου μνήμης που απαιτείται για την αποθήκευση του γράφου.

Οι δομές που χρησιμοποιούνται συνήθως για την αποθήκευση γράφων, είναι οι συνδεδεμένες λίστες ή λίστες γειτνίασης, οι δυδιάστατοι πίνακες και ορισμένες δομές δεδομένων που προκύπτουν ως συνδυασμός των δύο προαναφερθεισών δομών. Η δομή που κάθε φορά υιοθετείται για την αναπαράσταση ενός γράφου εξαρτάται, τόσο από τη δομή του γράφου όσο και από τον αλγόριθμο που εφαρμόζεται για το χειρισμό του γράφου.

Η συνδεδεμένη λίστα ή λίστα γειτνίασης (adjacency list), εφαρμόζεται συνήθως για την αποθήκευση αραιών και σχετικά μικρών γράφων (sparse graphs) όπου ο αριθμός των ακμών τους $|E|$ είναι κατά πολύ μικρότερος από τον αριθμό του τετραγώνου των κόμβων τους $|V^2|$. Σε κάθε κόμβο του γράφου, αντιστοιχεί μια λίστα η οποία περιλαμβάνει το σύνολο των κόμβων προς τους οποίους υπάρχει δυνατότητα άμεσης πρόσβασης από τον αρχικό κόμβο. Η λίστα ενός κόμβου ο οποίος ανήκει σε μη-κατευθυνόμενο γράφο περιλαμβάνει όλους τους κόμβους- γείτονες του αρχικού. Στην περίπτωση ενός κατευθυνόμενου γράφου με βάρη, η λίστα ενός κόμβου περιλαμβάνει το σύνολο των γειτονικών του κόμβων, που είναι προσβάσιμοι από τον αρχικό, συνοδευόμενων από το βάρος που αντιστοιχεί στην εκάστοτε παρεμβαλλόμενη ακμή. Η συνδεδεμένη λίστα ως δομή έχει μικρότερες απαιτήσεις μνήμης σε σχέση με τον πίνακα γειτνίασης, καθώς για ένα γράφο με K κόμβους και A ακμές οι απαιτήσεις σε χωρητικότητα είναι ίσες με $K+A$, ενώ συνιστά καταλληλότερη δομή για την απάντηση ερωτημάτων που σχετίζονται με την απαρίθμηση κόμβων του γράφου, όπως για παράδειγμα το ερώτημα: Βρες το σύνολο των γειτονικών κόμβων ενός κόμβου u [20], [3], [28].

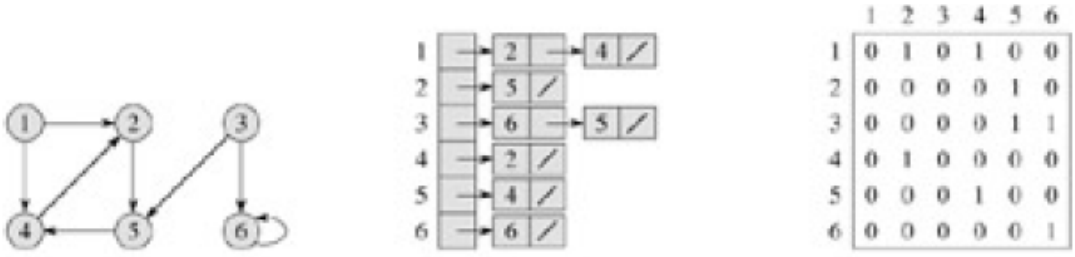
Ο δυδιάστατος πίνακας συνιστά την απλούστερη δομή που χρησιμοποιείται για την αναπαράσταση ενός γράφου. Οι γραμμές και οι στήλες του πίνακα περιλαμβάνουν τους κόμβους του γράφου, κάθε κόμβος δηλαδή περιγράφεται από μια γραμμή και μια στήλη. Στην περίπτωση ενός μη- κατευθυνόμενου γράφου, τα κελιά του πίνακα παίρνουν τιμές 1 ή 0, ανάλογα με το εάν υφίσταται ή όχι σύνδεση μεταξύ δύο κόμβων, του κόμβου γραμμής και του κόμβου στήλης. Ο πίνακας στην περίπτωση αυτή είναι συμμετρικός ως προς τα στοιχεία που περιλαμβάνονται στην κύρια διαγώνιο και είναι ίσα με 1. Η ίδια λογική ακολουθείται και κατά τη διαδικασία αναπαράστασης ενός κατευθυνόμενου γράφου, με τη διαφορά ότι στους κόμβους που περιλαμβάνονται στις γραμμές του πίνακα ανατίθεται ο ρόλος «από», ενώ στους κόμβους που περιλαμβάνονται στις στήλες του πίνακα ανατίθεται ο ρόλος «προς». Στην περίπτωση αναπαράστασης σταθμισμένου γράφου με δυδιάστατο πίνακα, τα πεδία του πίνακα περιλαμβάνουν τα βάρη των

αντίστοιχων ακμών, ενώ στην περίπτωση που μεταξύ δύο κόμβων δεν υφίσταται ακμή, στο αντίστοιχο κελί του πίνακα ανατίθεται τιμή ίση με το άπειρο *(∞)*. Ο δυδιάστατος πίνακας εφαρμόζεται συνήθως για την αναπαράσταση πυκνών γράφων όπου ο αριθμός των ακμών $|E|$ του γράφου είναι περίπου ίσος με τον αριθμό του τετραγώνου των κόμβων του $|V^2|$. Επιπρόσθετα, όταν απαιτείται γρήγορη απάντηση σε ερωτήματα του τύπου εάν υπάρχει μια ακμή που να συνδέει δύο οποιουσδήποτε κόμβους ενός γράφου, ο πίνακας προτιμάται ως δομή καθώς συνεπάγεται ταχύτερη προσπέλαση των στοιχείων που βρίσκονται αποθηκευμένα σε αυτόν, σε αντίθεση με τη λίστα, η οποία πρέπει να προσπελάσει τα αποθηκευμένα στοιχεία ένα προς ένα προκειμένου να δοθεί απάντηση στο ερώτημα που έχει τεθεί. Ωστόσο, ο πίνακας έχει μεγαλύτερες απαιτήσεις διαθέσιμης μνήμης οι οποίες για ένα γράφο που περιλαμβάνει K κόμβους μεταφράζονται σε απαιτήσεις χωρητικότητας ίσες με K^2. Τέλος, όταν ο γράφος δεν έχει βάρη, ο πίνακας ως δομή παρέχει ένα επιπλέον πλεονέκτημα που συνίσταται στο γεγονός ότι για κάθε είσοδο στον πίνακα απατείται μόνο ένα bit για την αναπαράστασή της [20], [3], [28].



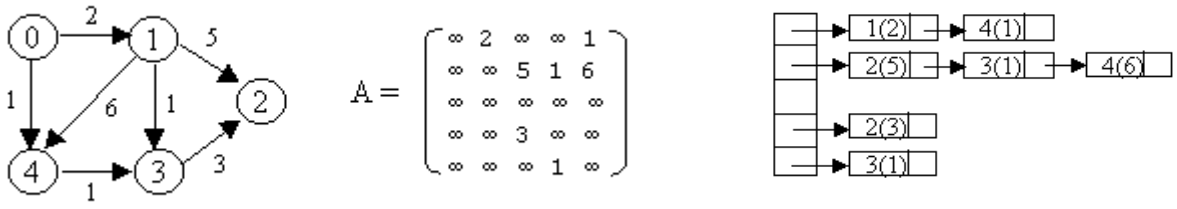
Σχήμα 2-1: Παράδειγμα Αναπαράστασης μη-Κατευθυνόμενου Γράφου σε Δομή Λίστας και Πίνακα

Πηγή: [3]



Σχήμα 2-2: Παράδειγμα Αναπαράστασης Κατευθυνόμενου Γράφου σε Δομή Λίστας και Πίνακα

Πηγή: [3]



Σχήμα 2-3: Παράδειγμα Αναπαράστασης Κατευθυνόμενου Γράφου με Βάρη σε Δομή Πίνακα και Λίστας

Πηγή: [URL21]

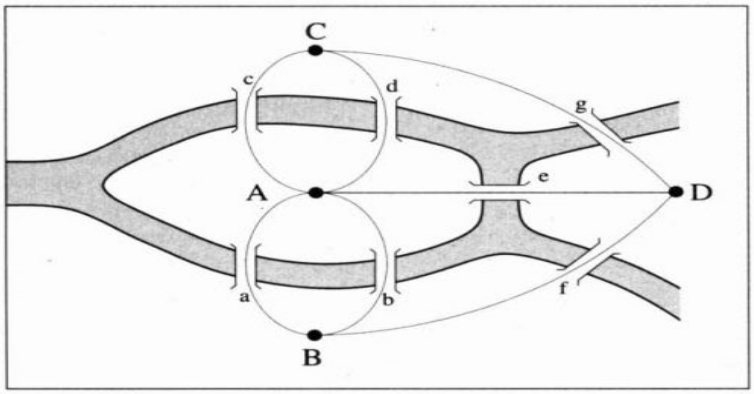
2.5 Μοντελοποίηση Προβλημάτων με Γράφους

Όπως έχει ήδη αναφερθεί σε προηγούμενο εδάφιο, παρά το γεγονός ότι τα θεμέλια της θεωρίας των γράφων τέθηκαν από τα μαθηματικά, οι βασικές αρχές της υιοθετούνται από διάφορα επιστημονικά πεδία προκειμένου να αναλυθούν και να ερευνηθούν προβλήματα, η επίλυση των οποίων απαιτεί τη μοντελοποίησή τους με τη βοήθεια γράφων. Οι βασικές κατηγορίες προβλημάτων η επίλυση των οποίων ανάγεται σε επίλυση γράφου, αφορούν προβλήματα όπου απαιτούνται επιλύσεις δικτύων και προβλήματα που σχετίζονται με τη μοντελοποίηση και διαγραμματική αναπαράσταση συστημάτων, αποτελούμενων από στοιχεία μεταξύ των οποίων υφίστανται κάποιου είδους σχέσεις αλληλεπίδρασης. Τα αντικείμενα αναπαρίστανται ως κόμβοι, ενώ οι συνδέσεις που υφίστανται μεταξύ τους ως ακμές.

Στο παρόν εδάφιο παρουσιάζονται ορισμένες περιπτώσεις προβλημάτων, τα οποία αναπαρίστανται διαγραμματικά ως γράφοι και η αναζήτηση των λύσεών τους ανάγεται στην επίλυση γράφου μέσα από την αξιοποίηση των βασικών αρχών της θεωρίας των γράφων.

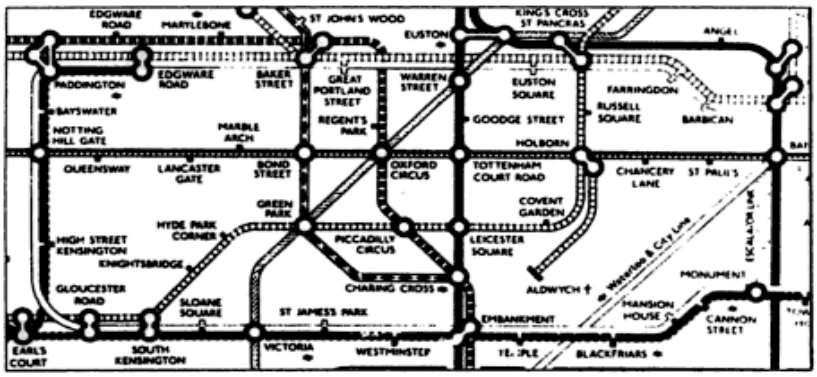
Το πρώτο ιστορικά πρόβλημα που παραπέμπει σε μοντελοποίηση δικτύου και επίλυση γράφου είναι αυτό που αφορούσε την περιστοιχιζόμενη από ποτάμια ύδατα πόλη του Koenigsberg, η οποία συνδεόταν μέσω επτά γεφυρών με την υπόλοιπη ενδοχώρα. Το πρόβλημα στην περίπτωση αυτή διατυπώθηκε ως ακολούθως: «Είναι δυνατό κάποιος να διασχίσει και τις επτά γέφυρες της πόλης επιστρέφοντας στο ίδιο σημείο από το οποίο ξεκίνησε διασχίζοντας την κάθε γέφυρα μία μόνο φορά;» και μπορεί να αναχθεί στη σχεδίαση των διαδρομών επάνω σε ένα φύλλο χαρτί με την προϋπόθεση ότι ο σχεδιαστής

δεν θα σηκώσει καμία φορά το μολύβι έως ότου ολοκληρωθεί η ζητούμενη διαδρομή. Η γραφική αναπαράσταση του προβλήματος παρουσιάζεται στο διάγραμμα που ακολουθεί [1].



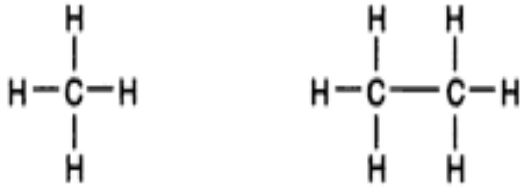
Διάγραμμα 2-8: Το Πρόβλημα των Επτά Γεφυρών του Koenigsberg
Πηγή: [13]

Η αναζήτηση της βέλτιστης διαδρομής που υφίσταται μεταξύ δύο σταθμών ενός υπόγειου σιδηρόδρομου ανάγεται σε επίλυση δικτύου που αναπαρίσταται ως γράφος. Η βέλτιστη διαδρομή αφορά συνήθως στη συντομότερη διαδρομή μεταξύ δύο σταθμών, σύμφωνα με κριτήρια που τίθενται ανά περίπτωση και μπορεί να αφορούν, τη γεωμετρική απόσταση μεταξύ των δύο σταθμών ή το χρόνο που απαιτείται για τη διάνυση της συγκεκριμένης διαδρομής. Στο διάγραμμα που ακολουθεί, παρουσιάζεται ο υπόγειος σιδηρόδρομος του Λονδίνου που έχει μοντελοποιηθεί ως γράφος αποτελούμενος από τους κόμβους- σταθμούς και τις ακμές- διαδρομές μεταξύ των σταθμών.



Διάγραμμα 2-9: Ο Υπόγειος Σιδηρόδρομος του Λονδίνου
Πηγή: [1]

Το επόμενο παράδειγμα προέρχεται από το χώρο της χημείας και τη διαγραμματική αναπαράσταση με τη μορφή γράφου της μοριακής δομής των αλκανίων, όπου τα άτομα του άνθρακα και του υδρογόνου αναπαρίστανται ως κόμβοι του γράφου ενώ οι χημικοί δεσμοί που υφίστανται μεταξύ των ατόμων αναπαρίστανται ως ακμές του γράφου. Η αναπαράσταση των μορίων με αυτόν τον τρόπο βοηθά στη μελέτη της δομής των μορίων, των χημικών δεσμών μεταξύ των ατόμων και της χημικής συμπεριφοράς των μορίων ως αποτέλεσμα της συγκεκριμένης χημικής δομής.



Διάγραμμα 2-10: Αναπαράσταση Μορίων Μεθανόλης και Αιθανόλης με τη βοήθεια Γράφου
Πηγή: [1]

Τα παραπάνω παραδείγματα συνιστούν ενδεικτικές περιπτώσεις προβλημάτων, η μελέτη των οποίων καθίσταται δυνατή μέσω της διαγραμματικής τους αναπαράστασης με τη βοήθεια ενός γράφου. Η θεωρία των γράφων, έχει σημαντικές εφαρμογές στην επίλυση γεωγραφικών προβλημάτων όπως το πρόβλημα εύρεσης συντομότερης διαδρομής, το πρόβλημα «του πλανόδιου πωλητή», η προσομοίωση του οδικού δικτύου μιας περιοχής, ο σχεδιασμός δικτύων κοινής ωφέλειας, η αναπαράσταση γειτονικών χωρών σε θεματικούς χάρτες κ.ά.

Στην παρούσα εργασία, διερευνάται το πρόβλημα εύρεσης συντομότερης διαδρομής σε οδικά δίκτυα μέσα από την ανάπτυξη μιας εφαρμογής στην οποία μοντελοποείται το βασικό οδικό δίκτυο της Αθήνας και αναπαρίσταται ως ένας κατευθυνόμενος γράφος με βάρη. Τα βάρη αφορούν εναλλακτικά αποστάσεις και χρόνους διάνυσης μιας διαδρομής. Στόχος είναι, η εξαγωγή αποτελεσμάτων που περιγράφουν τη συντομότερη διαδρομή μεταξύ δύο ή περισσοτέρων σημείων ανάλογα με τα κριτήρια που τίθενται κατά περίπτωση.

2.6 Γεωγραφικά Προβλήματα σε Γράφους

Οι γράφοι υιοθετούνται ως δομή για τη γραφική αναπαράσταση προβλημάτων, τα οποία αφορούν σε διακριτά αντικείμενα και στις σχέσεις που υφίστανται μεταξύ τους. Τέτοιου είδους προβλήματα είναι δυνατό να αναπαρασταθούν γραφικά ως κάποιας μορφής δίκτυα. Το γνωστικό υπόβαθρο και οι βασικές αρχές της θεωρίας των γράφων συνιστούν τα πλέον κατάλληλα μαθηματικά εργαλεία για την επίλυση προβλημάτων αυτής της κατηγορίας, που μπορεί να αναφέρονται σε διαφορετικά επιστημονικά πεδία.

Το παρόν εδάφιο εστιάζει στην παρουσίαση γεωγραφικών προβλημάτων, για την αναπαράσταση των οποίων έχουν υιοθετηθεί ως γραφικό υπόβαθρο οι γράφοι, ενώ για τη διαδικασία επίλυσής τους έχουν αναπτυχθεί αποτελεσματικοί αλγόριθμοι οι οποίοι περιγράφονται στη συνέχεια. Τα γεωγραφικά προβλήματα που περιγράφονται στο εδάφιο αυτό αφορούν κατά βάση δίκτυα υφιστάμενα στο χώρο τα οποία μοντελοποιούνται ως γράφοι.

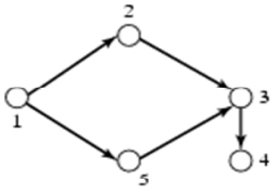
2.6.1 Διάσχιση γράφου

Το πρόβλημα της διάσχισης ενός γράφου (graph traversal) αφορά στην επίσκεψη των κόμβων του μέσω των ακμών που τους συνδέουν. Ουσιαστικά, περιλαμβάνει την αναζήτηση μονοπατιών μέσω των οποίων καθίσταται δυνατή η πρόσβαση σε έναν κόμβο προορισμού από έναν κόμβο αφετηρίας, διαμέσου άλλων κόμβων του γράφου ή την εύρεση μονοπατιών που διέρχονται από δύο ή περισσότερους κόμβους του γράφου. Τα περισσότερα γεωγραφικά προβλήματα γράφων συνιστούν ειδικές περιπτώσεις του προβλήματος διάσχισης γράφου και η επίλυσή τους ανάγεται στο πρόβλημα της διάσχισης γράφου στη βάση κάποιων κριτηρίων που τίθενται κατά περίπτωση.

2.6.2 Το πρόβλημα της μεταβατικής κλειστότητας

Ο προσδιορισμός της μεταβατικής κλειστότητας (transitive closure) ενός γράφου συνιστά ένα από τα σημαντικότερα προβλήματα γράφων, η επίλυση του οποίου δεν είναι δυνατή μόνο με την υιοθέτηση τεχνικών της σχεσιακής άλγεβρας. Αποτελεί τη βάση επάνω στην οποία στηρίζονται διάφορα άλλα προβλήματα που σχετίζονται με την ανάλυση δικτύων όπως το πρόβλημα εύρεσης συντομότερης διαδρομής, ο προσδιορισμός της συνδεσιμότητας γράφου κ.λπ. Η μεταβατική κλειστότητα G^* ενός γράφου $G\ (V, E)$

ορίζεται ως ένας γράφος που έχει ίσο αριθμό κόμβων με το γράφο *G* και κάθε ακμή του αντιστοιχεί σε ένα μονοπάτι του γράφου *G*. Συνεπώς, κάθε ακμή του γράφου *G** ανήκει στο σύνολο των ακμών του *E**, αν και μόνο αν αντιστοιχεί σε ένα μονοπάτι που συνδέει δύο κόμβους του γράφου *G*. Ως εκ τούτου, η εύρεση της μεταβατικής κλειστότητας ενός γράφου ανάγεται στην αναζήτηση μονοπατιών στον αρχικό γράφο και στην παραγωγή ενός νέου γράφου, οι ακμές του οποίου αποτελούνται από το σύνολο των ακμών του αρχικού γράφου συν τις νέες ακμές που προκύπτουν από την άμεση σύνδεση κόμβων, οι οποίοι στον αρχικό γράφο συνδέονται μέσω μονοπατιών [20]. Στο σχήμα που ακολουθεί, παρουσιάζεται η μεταβατική κλειστότητα *G** ενός γράφου *G*. Οι γραφικές αναπαραστάσεις των γράφων συνοδεύονται από δύο πίνακες, οι οποίοι περιλαμβάνουν τους κόμβους αφετηρίας και τους κόμβους προορισμού κάθε ακμής. Είναι προφανές ότι η μεταβατική κλειστότητα περιλαμβάνει μεγαλύτερο πλήθος ακμών σε σχέση με τον αρχικό γράφο καθώς έχουν προστεθεί οι ακμές που προκύπτουν από την άμεση σύνδεση κόμβων που στον αρχικό γράφο συνδέονται μέσω μονοπατιών.

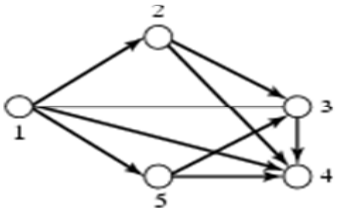


R

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3

Σχήμα 2-4: Ο Γράφος G

Πηγή: [20]



X

SOURCE	DEST
1	2
1	5
2	3
3	4
5	3
1	3
2	4
5	4
1	4

Σχήμα 2-5: Η Μεταβατική Κλειστότητα G του Γράφου G*

Πηγή: [20]

2.6.3 Το πρόβλημα εύρεσης της συντομότερης διαδρομής

Το πρόβλημα εύρεσης της συντομότερης διαδρομής (shortest path problem) αφορά στην εύρεση μιας διαδρομής μεταξύ δύο κόμβων *u* και *v* ενός γράφου όπου το κόστος της διαδρομής, η οποία περιλαμβάνει το σύνολο των ενδιάμεσων ακμών που συνδέουν τους δύο κόμβους, ελαχιστοποιείται σε σύγκριση με το κόστος όλων των εναλλακτικών διαδρομών μέσω των οποίων είναι δυνατή η μετάβαση από τον κόμβο *u* στον κόμβο *v*. Το κόστος μιας διαδρομής μπορεί να αναφέρεται είτε στην απόσταση μεταξύ των κόμβων αφετηρίας και προορισμού, είτε στο χρόνο μετάβασης από τον έναν κόμβο στον άλλο, είτε στο αντίτιμο που κάποιος οδηγός θα πρέπει να πληρώσει σε σταθμούς διοδίων όταν ο γράφος προσομοιώνει οδικά δίκτυα, είτε σε οποιαδήποτε άλλη παράμετρο υπεισέρχεται στα πλαίσια ανάλυσης της εκάστοτε εφαρμογής.

Στη μαθηματική του έκφραση το πρόβλημα εύρεσης συντομότερης διαδρομής σε γράφο διατυπώνεται ως εξής [3]:

Έστω ο σταθμισμένος γράφος *G (V, E)* όπου *V* το σύνολο των κόμβων του γράφου, *E* το σύνολο των ακμών του γράφου και $w : E \rightarrow R$ η συνάρτηση βάρους του γράφου. Το πρόβλημα που τίθεται, αφορά στην εύρεση του συντομότερου μονοπατιού μέσω του οποίου συνδέονται οι κόμβοι *u* και *v* του γράφου. Το βάρος ενός μονοπατιού $p = v_0, v_1, \ldots\ldots, v_k$ ισούται με το άθροισμα των βαρών των ακμών που συνθέτουν αυτό το μονοπάτι [3]. Είναι δηλαδή ίσο με:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

Το συντομότερο μονοπάτι *δ (u, v)* μεταξύ δύο κόμβων *u* και *v* είναι αυτό που ελαχιστοποιεί το βάρος διάνυσης της συγκεκριμένης διαδρομής και ικανοποιεί τις συνθήκες που φαίνονται στη σχέση που ακολουθεί. Η πρώτη σχέση ισχύει όταν υφίσταται μονοπάτι που συνδέει τους δύο κόμβους, ενώ η δεύτερη στην αντίθετη περίπτωση [3]:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} \\ \infty \end{cases}$$

Ανάλογα με την περίπτωση που κάθε φορά εξετάζεται, διακρίνονται τέσσερις υποπεριπτώσεις του προβλήματος εύρεσης συντομότερης διαδρομής οι οποίες διατυπώνονται ως εξής [3]:

- Το πρόβλημα εύρεσης συντομότερης διαδρομής μεταξύ ενός κόμβου αφετηρίας και ενός κόμβου προορισμού σε ένα γράφο (single- pair shortest path problem),
- Το πρόβλημα εύρεσης συντομότερης διαδρομής μεταξύ ενός κόμβου αφετηρίας και όλων των υπόλοιπων κόμβων του γράφου (single- source shortest path problem),
- Το πρόβλημα εύρεσης συντομότερης διαδρομής μεταξύ ενός κόμβου προορισμού και των υπόλοιπων κόμβων του γράφου (single- destination shortest path problem),
- Το πρόβλημα εύρεσης συντομότερης διαδρομής μεταξύ του συνόλου των ζευγών κόμβων που περιλαμβάνει ο γράφος (all- pairs shortest path problem).

Τέλος, θα πρέπει να σημειωθεί ότι σε μη σταθμισμένους γράφους, οι ακμές των οποίων θεωρούνται ισότιμες, η συντομότερη διαδρομή μεταξύ δύο κόμβων ορίζεται ως η διαδρομή εκείνη που περιλαμβάνει το μικρότερο πλήθος ακμών.

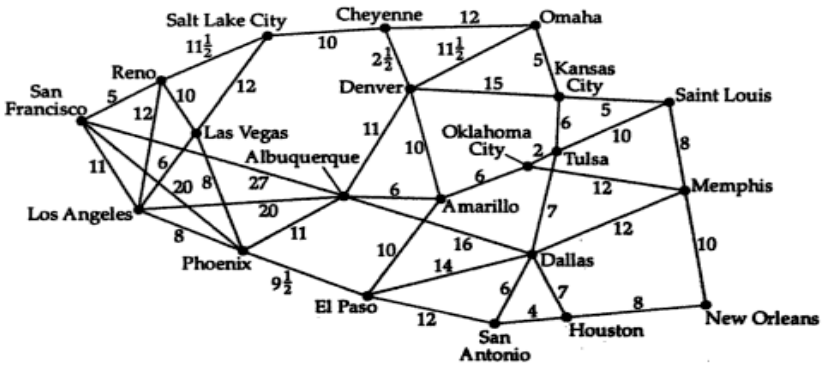
2.6.4 Το πρόβλημα του πλανόδιου πωλητή

Το πρόβλημα του πλανόδιου πωλητή αφορά στην εύρεση μιας διαδρομής που πρέπει να ακολουθήσει ένας πωλητής ο οποίος κινείται επάνω σε ένα δίκτυο πόλεων, ούτως ώστε να επισκέπτεται το σύνολο των πόλεων ελαχιστοποιώντας το κόστος της διαδρομής που ακολουθεί. Το δίκτυο των πόλεων αναπαρίσταται με ένα γράφο, οι κόμβοι του οποίου αναπαριστούν τις πόλεις ενώ οι ακμές του το οδικό δίκτυο που συνδέει τις πόλεις αυτές. Για την επίλυση του συγκεκριμένου προβλήματος, έχουν προταθεί διάφοροι αλγόριθμοι οι οποίοι ως ένα βαθμό επιλύουν ικανοποιητικά το πρόβλημα. Ωστόσο, η πολυπλοκότητα επίλυσης του προβλήματος, ιδιαίτερα όταν αυτό αφορά μεγάλους γράφους, είναι μη- πολυωνυμική καθώς η μόνη μέθοδος που εγγυάται τη λύση του προβλήματος συνίσταται στον εξαντλητικό υπολογισμό του κόστους (exhaustion method) του συνόλου των πιθανών διαδρομών και στην επιλογή της συντομότερης. Αυτό όμως καθίσταται εξαιρετικά πολύπλοκο για μεγάλους γράφους, διότι αυξάνεται πολύ ο χρόνος επεξεργασίας των δεδομένων. Συνεπώς, δεν υφίσταται μια συγκεκριμένη ενιαία μέθοδος που να λύνει οποιαδήποτε περίπτωση του προβλήματος του πλανόδιου πωλητή.

Υπάρχουν μόνο ad hoc διαδικασίες που βασίζονται σε ευρετικούς κανόνες οι οποίες όμως λύνουν προσεγγιστικά το πρόβλημα καθώς δε δίνουν τη βέλτιστη λύση [1], [28].

Μια ευρετική λύση του προβλήματος θα μπορούσε να διατυπωθεί ως ακολούθως: Έστω ένας κόμβος *ε*, ο οποίος επιλέγεται ως κόμβος εκκίνησης και ονομάζεται κόμβος *ν*. Εν συνεχεία, αναζητείται κατά προτεραιότητα η ακμή *(ν, u)* από τον κόμβο *ν* η οποία δεν έχει διαπεραστεί και έχει το ελάχιστο κόστος διάσχισης. Αφού, επισημανθεί ως διαπερασμένη, ο κόμβος *u* μετονομάζεται σε κόμβο *ν* και η διαδικασία επαναλαμβάνεται μέχρις ότου προσπελαθούν όλοι οι κόμβοι και ο αλγόριθμος επιστρέψει στο αρχικό σημείο εκκίνησης [28].

Στο σχήμα που ακολουθεί, παρουσιάζεται διαγραμματικά ένα δίκτυο πόλεων στις ΗΠΑ, το οποίο έχει μοντελοποιηθεί ως ένας σταθμισμένος γράφος και συνιστά μια περίπτωση δικτύου για την εφαρμογή αλγόριθμων επίλυσης του προβλήματος του πλανόδιου πωλητή.



Σχήμα 2-6: Δίκτυο Πόλεων στις Η.Π.Α.

Πηγή: [1]

2.6.5 Το πρόβλημα χρωματισμού πολυγώνων σε θεματικούς χάρτες

Το πρόβλημα του χρωματισμού πολυγώνων σε θεματικούς χάρτες αφορά στη χρήση του ελάχιστου δυνατού αριθμού χρωμάτων για το χρωματισμό γειτονικών πολυγώνων, έτσι ώστε να μην ανατεθεί σε γειτονικά μεταξύ τους πολύγωνα το ίδιο χρώμα. Το πρόβλημα αυτό μπορεί να μοντελοποιηθεί με τη βοήθεια ενός γράφου, οι κόμβοι του οποίου αναπαριστούν τα πολύγωνα και οι ακμές του, τη σχέση γειτνίασης που υφίσταται μεταξύ των πολυγώνων. Όπως και στην περίπτωση του προβλήματος του πλανόδιου

πωλητή, η επίλυση του προβλήματος χρωματισμού πολυγώνων απαιτεί την εφαρμογή μη-πολυωνυμικών αλγόριθμων. Ως εκ τούτου, εφαρμόζονται αλγόριθμοι ο σχεδιασμός των οποίων βασίζεται σε ευρετικές λύσεις οι οποίοι επιλύουν προσεγγιστικά το πρόβλημα χωρίς όμως να εξασφαλίζουν την εξαγωγή της βέλτιστης λύσης. Μια διατύπωση του προβλήματος στα πλαίσια μιας πραγματικής εφαρμογής θα μπορούσε να είναι η ακόλουθη: «Πόσα χρώματα απαιτούνται για το χρωματισμό των χωρών της Ευρώπης;». Στην περίπτωση αυτή, οι χώρες της Ευρώπης συνιστούν τους κόμβους ενός γράφου, οι γειτονικές χώρες συνδέονται μέσω των ακμών του γράφου, ενώ η λύση του προβλήματος δίνεται μέσα από την εφαρμογή ευρετικών αλγόριθμων [28].

2.7 Αλγόριθμοι Διάσχισης Γράφων

Στις παραγράφους που ακολουθούν, περιγράφονται μερικοί από τους κυριότερους αλγόριθμους διάσχισης γράφων, τα βασικά χαρακτηριστικά τους, η δομή και η λειτουργία τους. Ανάλογα με τις ιδιαιτερότητες και τις απαιτήσεις του εκάστοτε προβλήματος, εφαρμόζεται κατά περίπτωση ο αλγόριθμος που εξασφαλίζει την αποδοτικότερη διαχείρισή του. Αυτός είναι άλλωστε και ο λόγος που οι αλγόριθμοι διάσχισης γράφων εμφανίζονται διαφοροποιημένοι ως προς τα δεδομένα που κάθε φορά επεξεργάζονται και ως προς τις διαδικασίες επεξεργασίας των δεδομένων αυτών.

Με την έννοια του αλγόριθμου, εννοείται μια ορθά ορισμένη υπολογιστική διαδικασία, η οποία περιλαμβάνει μια τιμή ή ένα σύνολο τιμών ως δεδομένα εισόδου (input) από την επεξεργασία των οποίων προκύπτει κάποιο αποτέλεσμα (output). Ένας αλγόριθμος δηλαδή, συνίσταται στην ακριβή διατύπωση μιας σειράς βημάτων η εκτέλεση των οποίων οδηγεί στην εξαγωγή κάποιου αποτελέσματος [3]. Σύμφωνα με τους Aldous and Wilson, ένας αλγόριθμος αποτελείται από τέσσερα βασικά τμήματα: τα δεδομένα εισόδου (input data), μια λίστα που περιλαμβάνει διαδικασίες επεξεργασίας των δεδομένων εισόδου (ordered list of instructions), μια εντολή λήξης των διαδικασιών επεξεργασίας των δεδομένων (stop instruction) και τα δεδομένα εξόδου (output data). Ο αλγόριθμος συνιστά τη βάση για την επίλυση οποιουδήποτε προβλήματος σε υπολογιστικό περιβάλλον. Ουσιαστικά, περιλαμβάνει το σχεδιασμό της διαδικασίας επίλυσης ενός προβλήματος μέσα από το σαφή και ακριβή καθορισμό επιμέρους βημάτων που σχετίζονται με τη διαχείριση και την επεξεργασία των δεδομένων. Ο αλγόριθμος εκφράζεται είτε σε γλώσσα κατανοητή από τον άνθρωπο και μη – κατανοητή από τον υπολογιστή, είτε με τη βοήθεια ψευδοκώδικα. Η εισαγωγή και η λειτουργία του

σε υπολογιστικό περιβάλλον προϋποθέτει τη «μετάφραση» των επιμέρους βημάτων του αλγόριθμου σε μια γλώσσα προγραμματισμού, η οποία είναι κατανοητή από τον υπολογιστή.

Στη συνέχεια, παρουσιάζονται ορισμένοι βασικοί αλγόριθμοι που έχουν σχεδιαστεί προκειμένου να διαχειρίζονται προβλήματα διάσχισης γράφων και ζητήματα δρομολόγησης κινούμενων αντικειμένων επάνω σε δίκτυα. Οι αλγόριθμοι αυτής της κατηγορίας είναι ιδιαίτερα σημαντικοί σε εφαρμογές πλοήγησης, σχεδιασμού διαδρομών και διαχείρισης κυκλοφορίας.

2.7.1 Κατά πλάτος και κατά βάθος διάσχιση γράφου

Στο εδάφιο αυτό παρουσιάζονται δύο απλοί αλγόριθμοι διάσχισης γράφων, οι οποίοι συνιστούν τη βάση για το σχεδιασμό αλγόριθμων που εφαρμόζονται για την επίλυση προβλημάτων, τα οποία με τη σειρά τους αφορούν ειδικές περιπτώσεις διάσχισης γράφου. Οι αλγόριθμοι αυτοί αναφέρονται ως «Κατά Πλάτος Αναζήτηση» γράφου (Breadth- First Search) και «Κατά Βάθος Αναζήτηση» γράφου (Depth- First Search). Πρόκειται για δύο αλγόριθμους οι οποίοι δομήθηκαν υπό διαφορετική λογική, διαφοροποιώντας ουσιαστικά τη διαδικασία αναζήτησης σε γράφους. Κάθε ένας από αυτούς ενσωματώνει τα ιδιαίτερα πλεονεκτήματα και χαρακτηριστικά του, τα οποία καθορίζουν και την επιλογή εφαρμογής του αντίστοιχου αλγόριθμου ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής.

Κατά Πλάτος Διάσχιση Γράφου (Breadth- First Search)

Η κατά πλάτος αναζήτηση γράφου συνιστά έναν από τους απλούστερους αλγόριθμους που εφαρμόζονται κατά τη διαδικασία διάσχισης ενός γράφου, ενώ παράλληλα αποτελεί τη βάση για το σχεδιασμό εξειδικευμένων αλγόριθμων διάσχισης γράφων όπως για παράδειγμα ο αλγόριθμος Dijkstra, ο σχεδιασμός του οποίου στηρίζεται στην ιδέα της κατά πλάτος αναζήτησης γράφου.

Η ονομασία του αλγόριθμου της κατά πλάτος αναζήτησης (Breadth- First Search, BFS) γράφου, υποδηλώνει και τον τρόπο με τον οποίο εφαρμόζεται ο συγκεκριμένος αλγόριθμος αναζήτησης στους γράφους. Ο συγκεκριμένος αλγόριθμος «επεκτείνει» τα νοητά όρια μεταξύ των κόμβων που έχουν ανακαλυφθεί κατά τη διαδικασία της αναζήτησης και εκείνων που δεν έχουν ανακαλυφθεί ακόμη. Ο αλγόριθμος αναζητά τους κόμβους που πρόκειται να επισκεφθεί, εφαρμόζοντας μια κατά πλάτος τεχνική

αναζήτησης καθώς δίνει προτεραιότητα στους κόμβους που απέχουν απόσταση *k* από τον αρχικό κόμβο έναντι εκείνων που απέχουν απόσταση *k + 1*. Με άλλα λόγια, ο αλγόριθμος επισκέπτεται πρώτα τους άμεσους κόμβους- γείτονες του αρχικού και στη συνέχεια με την ίδια λογική συνεχίζει την αναζήτηση στους υπόλοιπους κόμβους του γράφου [3].

Δοθέντος ενός αρχικού κόμβου *u*, αναζητώνται όλοι οι κόμβοι του γράφου οι οποίοι είναι προσβάσιμοι από τον αρχικό κόμβο *u*. Κατά τη διάρκεια της αναζήτησης, ο αλγόριθμος επισκέπτεται πρώτα τους άμεσα γειτονικούς κόμβους του αρχικού, οι οποίοι βρίσκονται αποθηκευμένοι στη λίστα γειτνίασης του αρχικού κόμβου. Στη συνέχεια, αφού ολοκληρώσει την επίσκεψη του συνόλου των γειτονικών κόμβων του αρχικού, συνεχίζει με την επίσκεψη των γειτονικών κόμβων του κόμβου η πρόσβαση του οποίου από τον αρχικό γίνεται με το μικρότερο κόστος. Η διαδικασία επαναλαμβάνεται περιοδικά μέχρις ότου προσπελαθούν όλοι οι κόμβοι του γράφου.

Η εφαρμογή της κατά πλάτος αναζήτησης σε ένα γράφο $G = (V , E)$, όπου *V* οι κόμβοι του γράφου και *E* οι ακμές του, στοχεύει στη συστηματική αναζήτηση του γράφου, προκειμένου να βρεθούν οι κόμβοι που είναι προσβάσιμοι από έναν κόμβο που έχει οριστεί ως αρχικός κόμβος *s* (source node) κατά την εκκίνηση της διαδικασίας αναζήτησης. Παράλληλα, εκτελείται ο υπολογισμός της απόστασης κάθε κόμβου από τον αρχικό κόμβο αναφοράς, η τιμή της οποίας συνίσταται αφενός μεν σε ένα μη-σταθμισμένο γράφο, στον αριθμό των ακμών μέσω των οποίων ο αρχικός κόμβος συνδέεται με τους υπόλοιπους κόμβους του γράφου, αφετέρου δε σε ένα σταθμισμένο γράφο, στο συνολικό βάρος των ακμών που συνιστούν το μονοπάτι μεταξύ των δύο κόμβων. Στόχος της διαδικασίας είναι η ανεύρεση του συντομότερου μονοπατιού, της απόστασης δηλαδή που περιλαμβάνει τον κατά το δυνατό μικρότερο αριθμό ακμών που πρέπει ο αλγόριθμος να διασχίσει, προκειμένου να υλοποιηθεί η μετάβαση από τον κόμβο αναφοράς στον τελικό κόμβο ή της απόστασης όπου το κόστος διάσχισής της ελαχιστοποιείται στην περίπτωση σταθμισμένων γράφων. Το αποτέλεσμα της αναζήτησης της παραπάνω διαδικασίας είναι η εξαγωγή των συντομότερων μονοπατιών (shortest paths), μέσω των οποίων ο αρχικός κόμβος συνδέεται με τους υπόλοιπους κόμβους του γράφου. Ο αλγόριθμος εφαρμόζεται τόσο σε κατευθυνόμενους όσο και σε μη κατευθυνόμενους γράφους. Στην περίπτωση του BFS αλγόριθμου, ο γράφος αναπαρίσταται ως μια λίστα γειτνίασης τα στοιχεία της οποίας είναι οι γειτονικοί κόμβοι του εκάστοτε κόμβου που ανακαλύπτεται κατά τη διαδικασία αναζήτησης [3].

Η κατά πλάτος διάσχιση γράφου έχει ως αποτέλεσμα τη δημιουργία ενός κατά πλάτος δένδρου (breadth- first tree), το οποίο αρχικά αποτελείται μόνο από τη ρίζα του που είναι

ο αρχικός κόμβος *s* και επεκτείνεται κάθε φορά που ανακαλύπτεται ένας νέος κόμβος, ο οποίος προστίθεται στο δένδρο μαζί με την ακμή που τον συνδέει με τον πρόγονο κόμβο του [3]. Στη συνέχεια, παρουσιάζεται ο ψευδοκώδικας του αλγόριθμου BFS.

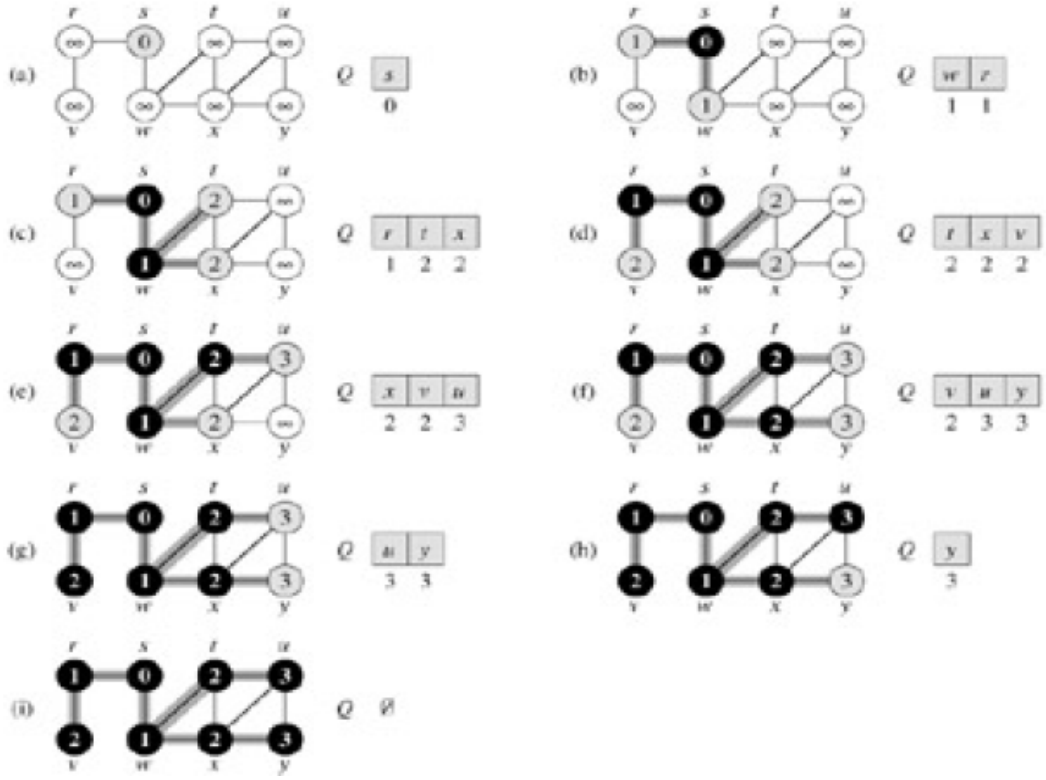
```
BFS (G , s)
1   for each vertex u ∈ V [G] - {s}
2        do color[u] ← WHITE
3           d[u] ← ∞
4           π[u] ← NIL
5   color[s] ← GRAY
6   d[s] ← 0
7   π[s] ← NIL
8   Q ← Ø
9   ENQUEUE(Q, s)
10  while Q ≠ Ø
11      do u ← DEQUEUE(Q)
12         for each v ∈ Adj[u]
13             do if color[v] = WHITE
14                   then color[v] ← GRAY
15                        d[v] ← d[u] + 1
16                        π[v] ← u
17                        ENQUEUE(Q, v)
18         color[u] ← BLACK
```

Σχήμα 2-7: Ο Ψευδοκώδικας του BFS Αλγόριθμου

Πηγή: [3]

Στις τέσσερις πρώτες γραμμές του ψευδοκώδικα, γίνεται η αρχικοποίηση των μεταβλητών και όλοι οι κόμβοι λαμβάνουν χρώμα λευκό το οποίο δηλώνεται μέσω της μεταβλητής *"color"* και σημαίνει ότι κανένας από τους κόμβους δεν έχει ακόμη προσπελαθεί από τον αλγόριθμο. Στη μεταβλητή *d[u]* που αντιστοιχεί στην απόσταση που υφίσταται μεταξύ του αρχικού κόμβου και των υπόλοιπων κόμβων του γράφου, δίνεται αρχικά τιμή ίση με το άπειρο για το σύνολο των κόμβων *u* του γράφου, ενώ στη μεταβλητή *π[u]* που αναπαριστά τον προκάτοχο ενός ανακαλυφθέντος κόμβου δίνεται αρχική τιμή ίση με το μηδέν. Στην πέμπτη γραμμή, ορίζεται ο αρχικός κόμβος αναφοράς ο οποίος λαμβάνει χρώμα γκρι, η απόσταση λαμβάνει την τιμή μηδέν καθώς δεν έχει ανακαλυφθεί κανένας άλλος κόμβος εκτός του αρχικού, ενώ η τιμή που αφορά στον προκάτοχο του ανακαλυφθέντος κόμβου είναι ίση με *NIL* διότι ο αρχικός κόμβος δεν έχει προκάτοχο κόμβο. Στις γραμμές 8 και 9 αρχικοποιείται η λίστα στην οποία καταχωρούνται οι κόμβοι που έχουν ανακαλυφθεί. Τέλος οι σειρές 10-18 περιλαμβάνουν

έναν επαναληπτικό βρόχο (loop) ο οποίος ξεκινά με την εντολή “while”. Οι εντολές που περιλαμβάνονται σε αυτό το τμήμα του κώδικα επαναλαμβάνονται όσο υπάρχουν γκρι κόμβοι στο γράφο, κόμβοι δηλαδή των οποίων οι λίστες γειτνίασης δεν έχουν ακόμη εξετασθεί πλήρως. Οι λίστες γειτνίασης «κρατούν» τους κόμβους οι οποίοι ανακαλύπτονται από τον αλγόριθμο σε σειρά προτεραιότητας. Ο πρώτος κόμβος που εισέρχεται στη λίστα είναι ο αρχικός κόμβος (source node), από τον οποίο ξεκινά η αναζήτηση στο γράφο. Κάθε κόμβος που ανακαλύπτεται από τον αλγόριθμο βγαίνει από τη λίστα όταν λάβει μαύρο χρώμα, γεγονός που σημαίνει ότι το σύνολο των γειτονικών του κόμβων έχει ανακαλυφθεί από τον αλγόριθμο αναζήτησης. Παράλληλα με τη διαδικασία ανεύρεσης κόμβων, ενημερώνονται οι μεταβλητές d και π που αφορούν στην απόσταση ενός ανακαλυφθέντος κόμβου από τον αρχικό και τους προκατόχους των ανακαλυφθέντων κόμβων αντίστοιχα. Η πολυπλοκότητα του συγκεκριμένου αλγόριθμου είναι ίση με $O (V + E)$, γεγονός που σημαίνει ότι ο χρόνος που απαιτείται για την ολοκλήρωση του αλγόριθμου μεταβάλλεται γραμμικά ως προς το μέγεθος της λίστας γειτνίασης που χρησιμοποιείται για την αναπαράσταση του γράφου. Από τον ψευδοκώδικα, καθίσταται πλήρως κατανοητό ότι η κατά πλάτος αναζήτηση γράφου αναζητά συντομότερες διαδρομές (shortest paths) μετάβασης από τον αρχικό κόμβο του γράφου προς το σύνολο των κόμβων οι οποίοι είναι προσβάσιμοι από τον αρχικό [3].



Διάγραμμα 2-11: Διαγραμματική Αναπαράσταση του Αλγόριθμου BFS

Πηγή: [3]

Κατά Βάθος Διάσχιση Γράφου (Depth- First Search)

Σε αντίθεση με τον αλγόριθμο που αφορά την κατά πλάτος διάσχιση γράφου, η κατά βάθος διάσχιση γράφου εστιάζει στη διαδοχική επέκταση του γράφου αναζητώντας κόμβους προσβάσιμους από τον αρχικό, σε διάφορα «βάθη». Έτσι, ο αλγόριθμος επισκέπτεται έναν άμεσο γείτονα του αρχικού κόμβου και στη συνέχεια προχωρά με τη διαδοχική εξερεύνηση του άμεσου γείτονα του κόμβου που μόλις ανακαλύφθηκε, ανεξάρτητα από το εάν έχει ανακαλυφθεί το σύνολο των γειτονικών κόμβων του αρχικού κόμβου, μέχρις ότου εξαντληθεί ένα ορισμένο βάθος. Στη συνέχεια, ο αλγόριθμος επιστρέφει στον κόμβο από τον οποίο ξεκίνησε την κατά βάθος αναζήτηση προκειμένου να εξερευνήσει άλλα βάθη, ξεκινώντας την αναζήτηση από έναν άλλο άμεσο γείτονα κόμβο του αρχικού. Η διαδικασία ολοκληρώνεται όταν ανακαλυφθούν όλοι οι κόμβοι που είναι προσβάσιμοι από τον αρχικό. Στην περίπτωση που μετά το πέρας της διαδικασίας αναζήτησης έχουν απομείνει κόμβοι που δεν έχουν ανακαλυφθεί, ορίζεται εκ

νέου ένας δεύτερος αρχικός κόμβος και η διαδικασία αναζήτησης επαναλαμβάνεται μέχρις ότου ανακαλυφθούν όλοι οι κόμβοι του γράφου [20], [3].

Όμοια με τον αλγόριθμο BFS, στην περίπτωση εφαρμογής του αλγορίθμου DFS, όταν ένας κόμβος ανακαλύπτεται, ο κόμβος που προηγείται του ανακαλυφθέντος και συνδέεται με αυτόν μέσω κοινής ακμής, ορίζεται ως ο προκάτοχος του ανακαλυφθέντος κόμβου και καταχωρείται στη μεταβλητή *π[u]*. Αρχικά, όλοι οι κόμβοι του γράφου λαμβάνουν χρώμα λευκό, όποιος κόμβος ανακαλύπτεται λαμβάνει χρώμα γκρι, ενώ όταν έχει ανακαλυφθεί το σύνολο των κόμβων που ανήκουν στη λίστα γειτνίασης του αρχικού κόμβου, τότε ο κόμβος αυτός λαμβάνει μαύρο χρώμα. Η κατά βάθος διάσχιση γράφου έχει ως αποτέλεσμα τη δημιουργία πολλαπλών κατά βάθος δένδρων (depth- first trees), καθώς η αναζήτηση δύναται να ξεκινά από περισσότερους του ενός αρχικών κόμβων. Το σύνολο των δένδρων του γράφου που προκύπτουν μετά την ολοκλήρωση του αλγόριθμου, συνιστούν ένα κατά βάθος δάσος (depth- first forest). Επιπλέον ο αλγόριθμος που εφαρμόζεται στην κατά βάθος αναζήτηση σημαίνει χρονικά κάθε κόμβο του γράφου. Κάθε κόμβος έχει δύο χρονικές σημάνσεις. Η πρώτη από αυτές αφορά την πρώτη φορά όπου ανακαλύφθηκε ο κόμβος και έλαβε γκρι χρώμα, ενώ η δεύτερη αφορά τη χρονική στιγμή όπου ο αλγόριθμος έχει ολοκληρώσει την αναζήτηση στο σύνολο των κόμβων που περιλαμβάνονται στη λίστα γειτνίασης του αρχικού, ο οποίος τότε λαμβάνει μαύρο χρώμα [3], [20]. Στο σχήμα 2-8 παρουσιάζεται ο ψευδοκώδικας της κατά βάθος αναζήτησης γράφου.

Οι τέσσερις πρώτες γραμμές του ψευδοκώδικα, περιλαμβάνουν την αρχικοποίηση των μεταβλητών. Όλοι οι κόμβοι του γράφου λαμβάνουν χρώμα λευκό και η μεταβλητή στην οποία καταχωρούνται οι προκάτοχοι κόμβοι *π[u]* των ανακαλυφθέντων κόμβων λαμβάνει αρχική τιμή *NIL*, δηλαδή κενή. Στην τέταρτη γραμμή, καταχωρείται η τιμή μηδέν στη μεταβλητή του χρόνου, ενώ στις γραμμές 5 έως 7 ελέγχεται το χρώμα του κόμβου και όταν αυτό είναι λευκό τότε ξεκινά μια διαδικασία κατά βάθος αναζήτησης από έναν αρχικό κόμβο. Κάθε φορά που καλείται αυτό το τμήμα του κώδικα, ο εκάστοτε κόμβος *u* του γράφου γίνεται η ρίζα ενός νέου δένδρου του κατά βάθος δάσους που δημιουργείται.

```
DFS (G)
1   for each vertex u ∈ V [G]
2        do color[u] ← WHITE
3           π[u] ← NIL
4   time ← 0
5   for each vertex u ∈ V [G]
6         do if color[u] = WHITE
7               then DFS-VISIT (u)
DFS-VISIT (u)
1  color[u] ← GRAY  ▶ White vertex u has just
been   discovered
2  time ← time + 1
3  d[u]  time
4  for each v ∈ Adj[u]  ▶ Explore edge (u,v)
5       do if color[v] = WHITE
6             then π[v] ← u
7                          DFS-VISIT(v)
8  color[u] ← BLACK    ▶ Blacken u; it is
finished
9  f[u] ▶ time ← time + 1
```

Σχήμα 2-8: Ο Ψευδοκώδικας του DFS Αλγόριθμου

Πηγή: [3]

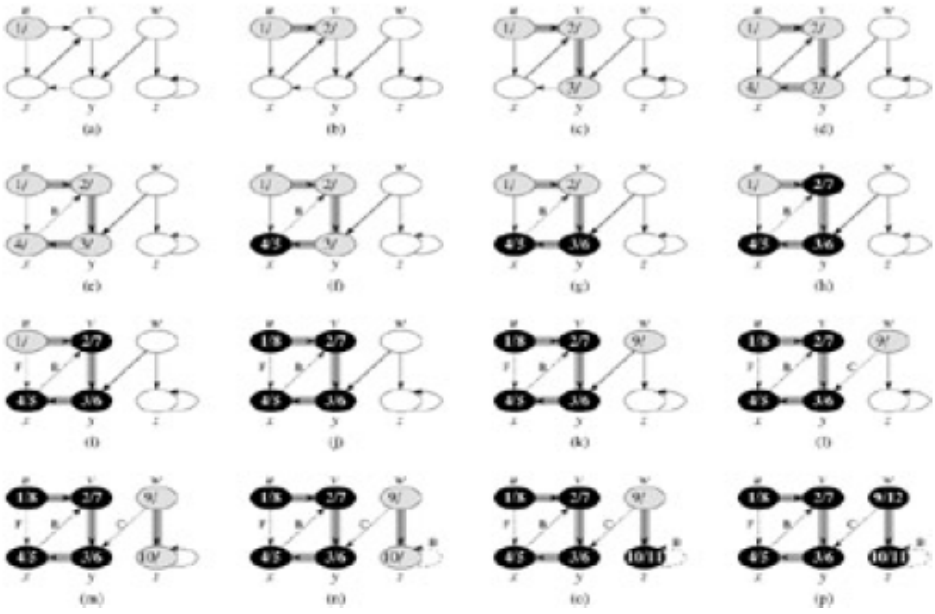
Όταν ο αλγόριθμος επιστρέφει στο σημείο αυτό κάθε κόμβος *u* έχει λάβει δύο χρονικές σημάνσεις, το χρόνο ανακάλυψης (discovery time) *d[u]* και το χρόνο περάτωσης της συγκεκριμένης αναζήτησης (finishing time) *f[u]*. Σε κάθε νέα κλήση του κώδικα, ο κόμβος *u* είναι αρχικά λευκός, τη στιγμή που ανακαλύπτεται λαμβάνει γκρι χρώμα, αυξάνεται η τιμή της μεταβλητής time και στη μεταβλητή *d[u]* καταχωρείται μια νέα χρονική στιγμή της ανακάλυψης του κόμβου. Ο κώδικας που περιλαμβάνεται στις γραμμές 4-7 του δεύτερου τμήματος του ψευδοκώδικα, ελέγχει κάθε γειτονικό κόμβο *v* του αρχικού κόμβου *u* και τον επισκέπτεται εάν αυτός είναι λευκός και έτσι ανακαλύπτεται και μια νέα ακμή του γράφου. Όταν ανακαλύπτεται το σύνολο των ακμών που συνδέονται με τον κόμβο *u*, τότε ο κόμβος *u* λαμβάνει χρώμα μαύρο και καταχωρείται η τιμή του χρόνου περάτωσης της συγκεκριμένης αναζήτησης στη μεταβλητή *f[u]* (γραμμές 8-9 του ψευδικώδικα) [3].

Τα αποτελέσματα μιας κατά βάθος αναζήτησης, εξαρτώνται από τη σειρά που ελέγχονται οι κόμβοι στη γραμμή 5 του DFS και από τη σειρά επισκεψιμότητας των γειτονικών κόμβων του εκάστοτε αρχικού κόμβου- σειρά 4 του DFS-VISIT. Τέλος, η πολυπλοκότητα του αλγόριθμου και ο χρόνος τρεξίματός του προκύπτει ως εξής: Η ολοκλήρωση των επαναλήψεων στις γραμμές 1-3 και 5-7 του DFS απαιτούν χρόνο

τρεξίματος ίσο με *Θ(V)*. Η διαδικασία που περιλαμβάνεται στο τμήμα του DFS-VISIT, καλείται ακριβώς μια φορά για κάθε κόμβο *v* ενώ ο βρόχος που περιλαμβάνεται στις γραμμές 4-7 εκτελείται |Adj[v]| φορές. Επομένως ισχύει η σχέση [3]:

$$\sum_{v \in V} |Adj[v]| = \Theta(E) .$$

Ως εκ τούτου, το κόστος εκτέλεσης του βρόχου είναι ίσο με *Θ(E)*. Συνεπώς, το συνολικό κόστος εκτέλεσης μιας κατά βάθος αναζήτησης γράφου είναι ίση με *Θ(V + E)*.



Διάγραμμα 2-12: Διαγραμματική Αναπαράσταση του Αλγόριθμου DFS
Πηγή: [3]

2.7.2 Ο αλγόριθμος του Dijkstra

Ο αλγόριθμος του Dijkstra επιλύει το πρόβλημα εύρεσης συντομότερης διαδρομής σε ένα γράφο από έναν κόμβο αφετηρίας προς έναν ή περισσότερους κόμβους προορισμού (single- source shortest path problem) σε ένα σταθμισμένο κατευθυνόμενο γράφο *G (V, E)* ο οποίος δεν περιλαμβάνει ακμές με αρνητικά βάρη, *w (u, v) ≥ 0* για κάθε ακμή *(u, v)*

που ανήκει στο σύνολο των ακμών *E* του γράφου. Επιπρόσθετα, επιλύει το πρόβλημα εύρεσης συντομότερης διαδρομής από έναν κόμβο προορισμού προς τους υπόλοιπους κόμβους του γράφου (single- destination shortest path problem), καθώς η αντιστροφή της κατεύθυνσης των ακμών του γράφου ανάγει την επίλυση του προβλήματος στην περίπτωση που αφορά την εύρεση συντομότερου μονοπατιού από έναν κόμβο αφετηρίας προς τους υπόλοιπους κόμβους του γράφου [3].

Ο αλγόριθμος, με δεδομένο έναν κόμβο αναφοράς, αναζητεί το ελάχιστο κόστος για τη μετάβαση από τον αρχικό κόμβο σε κάθε κόμβο του γράφου. Αρχικά, αναθέτει άπειρο κόστος μετάβασης προς όλους τους κόμβους του γράφου και στη συνέχεια δημιουργεί δύο υποσύνολα στα οποία κατανέμονται οι κόμβοι. Το πρώτο περιλαμβάνει τους κόμβους που ανακαλύπτονται διαδοχικά κατά την επίλυση και το οποίο αρχικά είναι κενό. Το δεύτερο αφορά γειτονικούς κόμβους των κόμβων που ανήκουν στο πρώτο σύνολο, οι οποίοι όμως δεν έχουν προσπελαθεί ακόμη. Παράλληλα, μια τρέχουσα μεταβλητή, φιλοξενεί τον κόμβο που ανήκει στο δεύτερο υποσύνολο με το ελάχιστο συσσωρευμένο κόστος. Η διαδικασία ολοκληρώνεται όταν προσπελαθούν όλοι οι κόμβοι του γράφου [28].

Αναλυτικότερα, ο αλγόριθμος, κρατά ένα σύνολο κόμβων *S* για τους οποίους έχουν προσδιοριστεί τα ελάχιστα συσσωρευμένα κόστη μετάβασης από έναν κόμβο αφετηρίας *s*. Μέσα από μια επαναληπτική διαδικασία, επιλέγει κάθε φορά έναν κόμβο *u*, ο οποίος ανήκει στο συντομότερο μονοπάτι που ξεκινά από τον κόμβο αφετηρίας και τον προσθέτει στο σύνολο *S*. Στη συνέχεια, δίνεται ο ψευδοκώδικας του αλγόριθμου του Dijkstra.

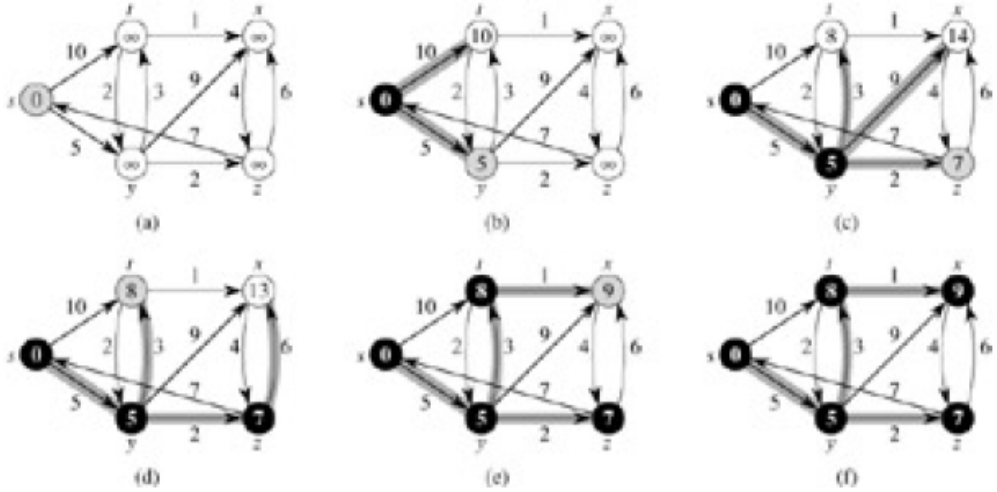
```
DIJKSTRA (G, W, S)
1   INITIALIZE-SINGLE-SOURCE(G,S)
2   S ← Ø
3   Q ← V[G]
4   while Q ≠ Ø
5       do u ← EXTRACT-MIN(Q)
6           S ← S ∈ {u}
7           for each vertex v ∈ Adj[u]
8              do RELAX(u, v, w)
```

Σχήμα 2-9: Ο Ψευδοκώδικας του Αλγόριθμου του Dijkstra

Πηγή: [3]

Στις πρώτες γραμμές 1-2 του ψευδοκώδικα, γίνεται η αρχικοποίηση των μεταβλητών. Στην αρχή κάθε επανάληψης που περιλαμβάνεται στο βρόχο *while* στις γραμμές 4-8, ο αλγόριθμος κρατά μια σταθερά *Q* η οποία είναι ίση με *Q = V - S*, δηλαδή ισούται με τη διαφορά των συνολικών κόμβων του γράφου μείον τους κόμβους που έχουν προσπελαθεί από τον αλγόριθμο και ανήκουν πλέον στο σύνολο κόμβων *S*, το οποίο περιλαμβάνει τους κόμβους που έχουν ανακαλυφθεί και για τους οποίους έχει υπολογιστεί το συντομότερο μονοπάτι από τον αρχικό κόμβο. Ο επαναληπτικός βρόχος τρέχει, μέχρις ότου το σύνολο κόμβων *Q* γίνει ίσο με το κενό. Αρχικά, όλοι οι κόμβοι του γράφου ανήκουν στο σύνολο *Q* ενώ κάθε φορά που ολοκληρώνεται μια επανάληψη αφαιρείται ένας κόμβος από το σύνολο *Q* και εισέρχεται στο σύνολο *S*. Όταν ένας κόμβος *u* εισέρχεται στο σύνολο *S*, ανήκει στο συντομότερο μονοπάτι που έχει ανακαλυφθεί μέχρι εκείνη τη συγκεκριμένη στιγμή και είναι ο κόμβος ο οποίος εξασφαλίζει το μικρότερο κόστος διάνυσης της συγκεκριμένης διαδρομής σε σχέση με όλους τους υπόλοιπους κόμβους του γράφου. Στις γραμμές 7-8 του ψευδοκώδικα, διερευνώνται όλες οι γειτονικές ακμές *(u, v)* του τρέχοντος κόμβου *u*, ενημερώνεται εκ νέου το κόστος διαδρομής *d[v]* και η μεταβλητή *π[v]* στην οποία καταχωρούνται οι προκάτοχοι κόμβοι στην περίπτωση όπου το συντομότερο μονοπάτι που οδηγεί στον κόμβο *v* μέσω του κόμβου *u*, είναι το βέλτιστο [3].

Η πολυπλοκότητα του αλγόριθμου του Dijkstra για ένα γράφο *G (V, E)* είναι ίση με $O(V^2)$. Είναι προφανές ότι όσο το πλήθος των ακμών του γράφου αυξάνεται, η πολυπλοκότητα του αλγόριθμου είναι απαγορευτική. Για τις περιπτώσεις αυτές, έχουν προταθεί αλγόριθμοι εύρεσης βέλτιστων διαδρομών που αξιοποιούν τις αρχές της τεχνητής νοημοσύνης. Ο πιο γνωστός αλγόριθμος αυτής της κατηγορίας, είναι ο *A**, τα χαρακτηριστικά του οποίου περιγράφονται στο εδάφιο που ακολουθεί.



Διάγραμμα 2-13: Διαγραμματική Αναπαράσταση του Αλγόριθμου του Dijkstra

Πηγή: [3]

2.7.3 Ο αλγόριθμος A*

Ο αλγόριθμος A* εφαρμόζεται για την επίλυση προβλημάτων εύρεσης συντομότερης διαδρομής από έναν κόμβο αφετηρίας προς έναν κόμβο προορισμού (single- pair shortest path problem). Η λογική βάσει της οποίας δομείται, είναι περίπου όμοια με τη λογική δόμησης του αλγόριθμου του Dijkstra. Σε αντίθεση όμως με τον αλγόριθμο του Dijkstra, χρησιμοποιείται σε περιπτώσεις μεγάλων γράφων όπου η πολυπλοκότητα του αλγόριθμου του Dijkstra καθίσταται απαγορευτική. Ο αλγόριθμος A* αξιοποιεί τεχνικές της τεχνητής νοημοσύνης, προκειμένου να αυξηθεί η ταχύτητα αναζήτησης στους γράφους μέσα από την εφαρμογή ευρετικών κανόνων (heuristics).

Αρχικά, εκτιμάται το κόστος μετάβασης από τον κόμβο αφετηρίας στον κόμβο προορισμού. Πρόκειται για μια ενδεικτική και όχι για μια ακριβή εκτίμηση. Στη συνέχεια, υπολογίζονται αναδρομικά τα κόστη μετάβασης στους γειτονικούς κόμβους του αρχικού κόμβου και αποκλείονται εξ αρχής μονοπάτια που έχουν μεγαλύτερο κόστος σε σχέση με αυτό που είχε αρχικά εκτιμηθεί [28].

Αναλυτικότερα, ο αλγόριθμος χρησιμοποιεί μια συνάρτηση *f (u, d)* εκτίμησης του κόστους μετάβασης από τον κόμβο αφετηρίας στον κόμβο προορισμού προκειμένου να εκτιμηθεί ένα αρχικό κόστος, στη βάση του οποίου πρόκειται να ελεγχθούν τα κόστη των μονοπατιών που ανακαλύπτονται στη συνέχεια. Ο ψευδοκώδικας του αλγόριθμου A* που παρουσιάζεται στο σχήμα που ακολουθεί, προκύπτει από τον ψευδοκώδικα του

αλγόριθμου του Dijkstra και εμφανίζεται διαφοροποιημένος αφενός ως προς τη συνάρτηση *f (u, d)* που χρησιμοποιείται για την αρχική εκτίμηση του κόστους και αφετέρου ως προς τους ευρετικούς κανόνες που υπεισέρχονται κατά τη διαδικασία εύρεσης του συντομότερου μονοπατιού [20].

```
1 procedure D i j k s t r a (G (V, E), v, d, f);
2 {
3   var: integer;
4     foreach u in V do {C (v, u) = inf; C(v, v) = 0; path
(v, u) :=null}
5    frontierSet := [v]; exploredSet :=emptySet;
6    while not_empty(frontierSet) do
7    {    select w from frontierSet with minimum (C (v, w)
+ f(w, d));
8         frontierSet := frontierSet – [w]; exploredSet :=
exploredSet + [w];
9         if (u = d) then terminate
10        else { fetch( w.adjacencyList);
11             foreach <  u, C(w, u) > in w.adjacencyList
12             if C(v, u) > C(v, w) + C(w, u) then
13             {
14                  C(v, u) := C(v, w) + C(w, u);
15                  path (v, u) := path (v, w) + (w, u);
16                  if u ϶ frontierSet U exploredSet then
17                  frontierSet := frontierSet + [u];
18             }
19        }
20   }
21 }
```

*Σχήμα 2-10: Ο Ψευδοκώδικας του Αλγόριθμου A**

Πηγή: [20]

Η διαδικασία αναζήτησης του αλγόριθμου A* ολοκληρώνεται μετά την επανάληψη εκείνη του βρόχου *while* όπου προσπελαύνεται ο κόμβος *u* ως ο κόμβος προορισμού της διαδρομής με το μικρότερο κόστος και εισάγεται στο σύνολο των επισκεφθέντων κόμβων *frontierSet* με το μικρότερο συσσωρευμένο κόστος. Όσο μικρότερο είναι το πλήθος των ακμών που συνθέτουν το συντομότερο μονοπάτι από τον κόμβο εκκίνησης στον κόμβο προορισμού, τόσο μικρότερος είναι ο χρόνος που απαιτείται για το τρέξιμο του αλγόριθμου. Ο αλγόριθμος δεν είναι απαραίτητο να επισκεφθεί το σύνολο των κόμβων του γράφου προκειμένου να ανακαλύψει τη συντομότερη διαδρομή μεταξύ των κόμβων αφετηρίας και προορισμού, καθώς πολλοί από αυτούς ανήκουν σε μονοπάτια τα οποία έχουν ήδη αποκλειστεί, πριν αυτοί οι κόμβοι προσπελαθούν λόγω υψηλού κόστους [20].

Η πολυπλοκότητα του αλγόριθμου Α* εξαρτάται από τους ευρετικούς κανόνες που χρησιμοποιούνται κατά περίπτωση. Έτσι, εάν η συνάρτηση εκτίμησης που χρησιμοποιεί ο αλγόριθμος είναι η *f(x)* τότε η πολυπλοκότητα του αλγόριθμου ορίζεται ως εξής:

$$|f(x) - f^*(x)| = O(\log f^*(x))$$

Όπου f^* είναι ο βέλτιστος ευρετικός κανόνας που χρησιμοποιεί ο αλγόριθμος για την εύρεση της συντομότερης διαδρομής σε ένα γράφο. Ο αλγόριθμος Α* είναι κατά κανόνα ταχύτερος από τον αλγόριθμο του Dijkstra.

2.7.4 Ο αλγόριθμος επίλυσης του προβλήματος του πλανόδιου πωλητή

Το πρόβλημα του πλανόδιου πωλητή συνιστά ένα από τα πολυπλοκότερα προβλήματα γράφων για το οποίο, δεν έχει ακόμη προταθεί μια αποτελεσματική ενιαία μέθοδος επίλυσης. Ο γράφος (σταθμισμένος και μη κατευθυνόμενος) υιοθετείται σε αυτήν την περίπτωση, για την προσομοίωση ενός δικτύου πόλεων οι οποίες συνδέονται μεταξύ τους μέσω του οδικού δικτύου. Το πρόβλημα που τίθεται, έγκειται στην εύρεση του αποτελεσματικότερου τρόπου διάσχισης του οδικού δικτύου από έναν πλανόδιο πωλητή, ο οποίος ξεκινά από μια πόλη και πρέπει να επισκεφθεί το σύνολο των υπόλοιπων πόλεων του δικτύου, περνώντας από κάθε πόλη το πολύ μια φορά και να επιστρέψει στο σημείο εκκίνησης. Ουσιαστικά αναζητείται η βέλτιστη διαδρομή, η διαδρομή δηλαδή με το ελάχιστο κόστος που πρέπει να διασχίσει ο πλανόδιος πωλητής μέχρι να επιστρέψει στο σημείο από το οποίο ξεκίνησε.

Τα δεδομένα βάσει των οποίων ξεκινά ο σχεδιασμός του αλγόριθμου, είναι ο αριθμός των πόλεων που πρέπει να επισκεφθεί ο πωλητής και το κόστος διάσχισης κάθε τμήματος του οδικού δικτύου που συνδέει τις πόλεις αυτές. Υποθέτοντας ότι ο συνολικός αριθμός των πόλεων που πρέπει να επισκεφθεί ο πλανόδιος πωλητής είναι ίσος με *n*, η πόλη *1* είναι η πόλη από την οποία ξεκινά τη διαδρομή του και *c (i, j)* το κόστος μετάβασης από την πόλη *i* στην πόλη *j*, τα βασικά βήματα του αλγόριθμου επίλυσης του προβλήματος είναι συνοπτικά τα ακόλουθα:

- Εύρεση του συνόλου των πιθανών λύσεων του προβλήματος ο αριθμός των οποίων για ένα σύνολο *n* πόλεων ανέρχεται στις *(n – 1)!* λύσεις.
- Προσδιορισμός του κόστους διάσχισης κάθε μίας από τις διαδρομές που βρέθηκαν στο πρώτο βήμα του αλγόριθμου.

- Επιλογή της διαδρομής με το ελάχιστο κόστος διάσχισης.

Είναι προφανές, ότι η ολοκλήρωση της παραπάνω διαδικασίας απαιτεί *(n – 1)!* βήματα όπου εξετάζεται η ύπαρξη κάθε πιθανής διαδρομής που δύναται να ακολουθήσει ο πλανόδιος πωλητής. Η διαδικασία ολοκληρώνεται θεωρητικά με την εύρεση *(n – 1)!* εναλλακτικών λύσεων. Ωστόσο στην πράξη, η αύξηση της πολυπλοκότητας του γράφου που προσομοιώνει το δίκτυο πόλεων καθιστά την παραπάνω διαδικασία επίλυσης του προβλήματος απαγορευτική, λόγω των εξαιρετικά μεγάλων χρονικών απαιτήσεων που προϋποθέτει η εξεύρεση του συνόλου των εναλλακτικών λύσεων. Αν για παράδειγμα το προς επίλυση δίκτυο αποτελείτο από *21* πόλεις, η διαδικασία εύρεσης των πιθανών λύσεων του προβλήματος θα απαιτούσε *(n – 1)! = (21 – 1)! = 20!* βήματα έως ότου ολοκληρωθεί. Εάν ο χρόνος που απαιτείται για την ολοκλήρωση κάθε βήματος ήταν ίσος με *1 msec* ο χρόνος υπολογισμού της βέλτιστης διαδρομής θα ήταν ίσος με *770 αιώνες*. Ως εκ τούτου, ο υπόλογισμός της βέλτιστης διαδρομής μέσα από τον έλεγχο του συνόλου των πιθανών λύσεων του προβλήματος είναι πρακτικά αδύνατος.

Η αδυναμία εφαρμογής μιας ενιαίας μεθόδου που να δίνει λύση στο πρόβλημα του πλανόδιου πωλητή, οδήγησε στην αναζήτηση ad hoc προσεγγίσεων με εφαρμογή ευρετικών κανόνων, οι οποίες εξασφαλίζουν την εξαγωγή προσεγγιστικών λύσεων του προβλήματος. Στη συνέχεια, παρουσιάζεται ενδεικτικά μια λύση που έχει προταθεί για το πρόβλημα του πλανόδιου πωλητή και αξιοποιεί ευρετικούς κανόνες.

Έστω λοιπόν ότι κάθε φορά που ο πλανόδιος πωλητής βρίσκεται σε μια πόλη *i* επιλέγει να επισκεφθεί την επόμενη πόλη *j* με κόστος διάσχισης της συγκεκριμένης διαδρομής ίσο με *c (i, j)*, το οποίο είναι το ελάχιστο κόστος σε σύγκριση με κάθε άλλο κόστος *c (i, k)* που προϋποθέτει η μετάβαση του πωλητή σε μια οποιαδήποτε άλλη πόλη *k* του γράφου, που ο πωλητής δεν έχει ακόμη επισκεφθεί. Σύμφωνα με τη λογική αυτή, προκύπτει ένας αλγόριθμος επίλυσης του προβλήματος (greedy algorithm) όπου ο πωλητής επιλέγει την επόμενη πόλη που πρόκειται να επισκεφθεί, με μοναδικό κριτήριο το ελάχιστο κόστος μετάβασης από τη μια πόλη στην άλλη, ανεξαρτήτως εάν αυτή η επαναλαμβανόμενη διαδικασία δίνει τελικά ένα αποτέλεσμα το οποίο δε συμπίπτει με τη βέλτιστη διαδρομή. Τα δεδομένα εισόδου που απαιτεί το τρέξιμο του παραπάνω αλγόριθμου, είναι ο συνολικός αριθμός των πόλεων *n* του δικτύου και το κόστος μετάβασης *c (i, j)* από μία πόλη *i* σε μια άλλη πόλη *j*. Το αποτέλεσμα που αναμένεται να δώσει ο αλγόριθμος, συνίσταται στη διανυσματική απεικόνιση της διαδρομής που πρέπει εν τέλει να ακολουθήσει ο πλανόδιος πωλητής και το συνολικό κόστος διάσχισης της διαδρομής. Το τρέξιμο του αλγόριθμου ξεκινά από την πόλη *1*. Το πρώτο βήμα

περιλαμβάνει την αρχικοποίηση των μεταβλητών, στο δεύτερο βήμα ορίζεται ο βρόχος επανάληψης βάσει του οποίου επισκέπτονται οι κόμβοι του γράφου και τέλος, στο τρίτο βήμα ορίζεται το συνολικό κόστος της διαδρομής που βρίσκει ο αλγόριθμος [URL4]. Ο ψευδοκώδικας του αλγόριθμου, παρουσιάζεται στο σχήμα 2-11 που ακολουθεί.

Ουσιαστικά, «προτείνεται» μια λύση του προβλήματος του πλανόδιου πωλητή με εφαρμογή ευρετικών κανόνων, ορίζεται ένας δείκτης *e* στον οποίο καταχωρούνται οι κόμβοι που έχει επισκεφθεί ο αλγόριθμος και μια συνάρτηση *C* για τον υπολογισμό του τελικού κόστους διαδρομής. Το τρέξιμο του αλγόριθμου ολοκληρώνεται, όταν η μεταβλητή *r* του βρόχου *for* γίνει ίση με τον αριθμό των πόλεων του δικτύου *n*, όταν δηλαδή έχουν προσπελαθεί όλοι οι κόμβοι του γράφου.

```
1 INITIALIZATION
  • c ← 0
  • Cost ← 0
  • visits ← 0
  • e = 1 /*pointer of the visited city*/
2   For 1 ≤ r ≤ n
        Do {
            Choose pointer j with
  • minimum = c(e,j)= min{c(e,k);visits(k) = 0 and 1 ≤ k ≤ n
    }
  • cost ← cost + minimum – cost
  • e = j
3 C(r) ← j
   • C(n) = 1
   • Cost = cost + c(e,1)
```

Σχήμα 2-11: Ο Ψευδοκώδικας του Αλγόριθμου του Πλανόδιου Πωλητή
Πηγή: [URL4]

2.7.5 Ο αλγόριθμος Bellman- Ford

Ο αλγόριθμος Bellman- Ford εφαρμόζεται κατά κύριο λόγο σε σταθμισμένους κατευθυνόμενους γράφους, για την επίλυση του προβλήματος εύρεσης συντομότερης διαδρομής από έναν κόμβο αφετηρίας προς έναν κόμβο προορισμού (single- source shortest path), στην περίπτωση που ένας γράφος περιλαμβάνει ακμές που έχουν θετικά και αρνητικά βάρη. Ο αλγόριθμος επιστρέφει μια λογική τιμή (boolean value), η οποία

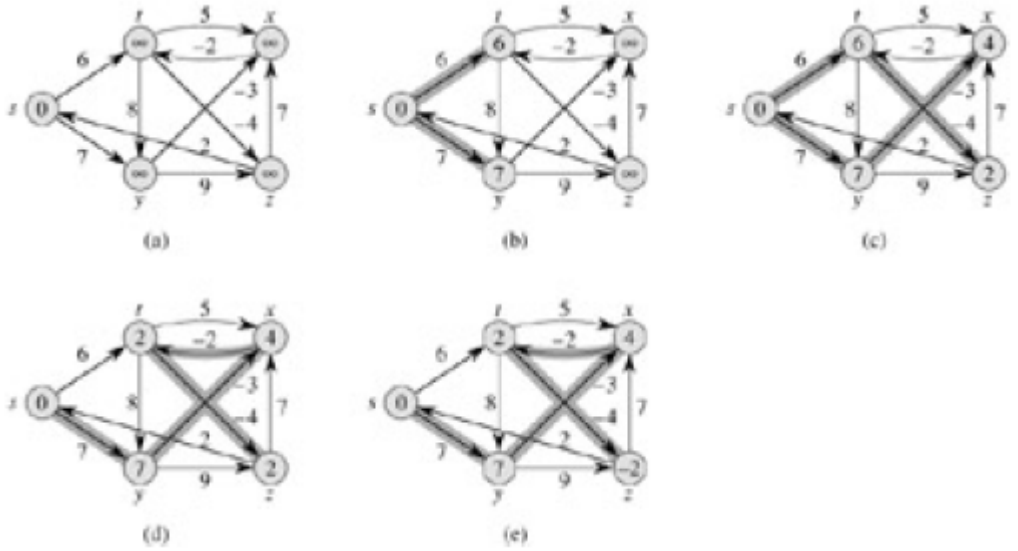
υποδεικνύει εάν υφίσταται ή όχι ένας αρνητικού βάρους κύκλος στο γράφο που είναι προσβάσιμος από τον κόμβο αφετηρίας. Η ύπαρξη ενός τέτοιου κύκλου στο γράφο σημαίνει ότι το πρόβλημα που εξετάζεται δεν έχει λύση. Εάν δεν υπάρχει κύκλος αρνητικού βάρους, τότε ο αλγόριθμος δίνει ως αποτελέσμα τις συντομότερες διαδρομές που υφίστανται μεταξύ των κόμβων του γράφου και τα κόστη διάσχισής τους. Η πολυπλοκότητα του αλγόριθμου Bellman – Ford σε ένα γράφο με *V* κόμβους και *E* ακμές είναι ίση με *O (V, E)* [3]. Ο ψευδοκώδικας του αλγόριθμου φαίνεται στο σχήμα που ακολουθεί.

```
BELLMAN-FORD (G, W, S)
1   INITIALIZE-SINGLE-SOURCE(G,S)
2   for i ← 1 to |V[G]| - 1
3       do for each edge (u,v) ∈ E[G]
4             do RELAX(u, v, w)
5   for each edge (u,v) ∈ E[G]
6          do if d[v] > d[u] + w(u,v)
7                then return FALSE
8   return TRUE
```

Σχήμα 2-12: Ο Ψευδοκώδικας του Αλγόριθμου Bellman - Ford

Πηγή: [3]

Στην πρώτη γραμμή του ψευδοκώδικα, γίνεται η αρχικοποίηση των μεταβλητών *d* και *π* στις οποίες καταχωρούνται η απόσταση που διανύεται κατά τη διάρκεια τρεξίματος του αλγόριθμου και οι προκάτοχοι κόμβοι αντίστοιχα. Ο αλγόριθμος, υλοποιεί *|V| - 1* περάσματα από τις ακμές του γράφου, κάθε ένα από τα οποία αντιστοιχεί σε μια επανάληψη του βρόχου που περιλαμβάνεται στην εντολή *for* στις γραμμές 2-4 του ψευδοκώδικα. Αφού ολοκληρωθούν τα *|V| -1* περάσματα από τις ακμές του γράφου, ο αλγόριθμος ελέγχει την ύπαρξη κύκλων με αρνητικά βάρη και επιστρέφει την αντίστοιχη boolean τιμή. Εάν υπάρχει κύκλος με αρνητικό βάρος το τρέξιμο του αλγόριθμου σταματά, σε διαφορετική περίπτωση ο αλγόριθμος δίνει ως αποτέλεσμα τα συντομότερα μονοπάτια και το κόστος τους.



Διάγραμμα 2-14: Διαγραμματική Αναπαράσταση του Αλγόριθμου Bellman - Ford
Πηγή: [3]

Στις παραπάνω παραγράφους, παρουσιάστηκαν οι κυριότεροι αλγόριθμοι που εφαρμόζονται για την επίλυση του προβλήματος εύρεσης συντομότερης διαδρομής μεταξύ δύο κόμβων ενός γράφου ή μεταξύ ενός αρχικού και του συνόλου των υπόλοιπων κόμβων. Στην περίπτωση που αναζητώνται συντομότερα μονοπάτια μεταξύ του συνόλου των κόμβων του γράφου (all pairs shortest path), εφαρμόζονται αλγόριθμοι ο σχεδιασμός των οποίων στηρίζεται σε μεθόδους δυναμικού προγραμματισμού. Στις περιπτώσεις αυτές, το πρόβλημα «σπάζει» σε μικρότερα υπο- προβλήματα τα οποία επιλύονται ξεχωριστά και η τελική ενιαία λύση που επιλύει το πρόβλημα συνολικά, προκύπτει ως μια σύνθεση των επιμέρους λύσεων. Στους αλγόριθμους αυτής της κατηγορίας, ανήκει ο αλγόριθμος Floyd- Warshall ο οποίος εφαρμόζεται σε σταθμισμένους κατευθυνόμενους γράφους με αρνητικά βάρη και ο αλγόριθμος Johnson's, ο οποίος εφαρμόζεται σε σχετικά αραιούς γράφους, είναι συντομότερος από τον αλγόριθμο Floyd- Warshall και χρησιμοποιεί ως υπορουτίνες τους αλγόριθμους Dijkstra και Bellman – Ford.

2.8 Βελτιώσεις στο Σχεδιασμό Αλγόριθμων Γράφων και Κόστος I/O

Η πολυπλοκότητα των προβλημάτων η επίλυση των οποίων ανάγεται στην επίλυση προβλημάτων σε γράφους, καθώς και οι απαιτήσεις που αφορούν στην ταχύτητα προσπέλασης των δεδομένων και την ταχύτητα εξαγωγής αποτελεσμάτων, καθιστά

απαραίτητη τη διαρκή βελτίωση των ήδη υφιστάμενων τεχνικών ή το σχεδιασμό νέων, βελτιωμένων και αποδοτικότερων αλγόριθμων.

Τα στοιχεία που καταχωρούνται σε μια χωρική βάση δεδομένων βρίσκονται αποθηκευμένα στη δευτερεύουσα μνήμη του υπολογιστή, για λόγους που σχετίζονται με τις αυξημένες απαιτήσεις σε χωρητικότητα. Ως εκ τούτου, κατά τη διαδικασία αναζήτησης μονοπατιών σε γράφους, τα δεδομένα που αφορούν το γράφο και τίθενται προς επεξεργασία από τον αλγόριθμο που εφαρμόζεται κάθε φορά, μεταφέρονται από τη δευτερεύουσα μνήμη στην οποία βρίσκονται αποθηκευμένα στην κύρια μνήμη του υπολογιστή. Συνεπώς, κατά το σχεδιασμό αλγόριθμων δρομολόγησης, μεγάλη έμφαση δίνεται στην ελαχιστοποίηση του *I/O* κόστους προκειμένου οι διαδικασίες διάσχισης γράφων να γίνονται κατά το δυνατό πιο αποδοτικές και αποτελεσματικές [20].

Στο εδάφιο αυτό, παρουσιάζονται συνοπτικά οι τάσεις που επικρατούν στο πεδίο του σχεδιασμού αλγόριθμων που τρέχουν σε γράφους και τα βασικά σημεία στα οποία εστιάζει η έρευνα για την ανάπτυξη βελτιωμένων αλγόριθμων και τεχνικών.

2.8.1 Ιεραρχικοί αλγόριθμοι

Οι ιεραρχικοί αλγόριθμοι, συνιστούν μια ιδιαίτερη κατηγορία αλγόριθμων ως προς τη λογική που εφαρμόζουν κατά τη διαδικασία αναζήτησης συντομότερων μονοπατιών σε γράφους. Οι αλγόριθμοι αυτής της κατηγορίας, διασπούν το γράφο σε επιμέρους τμήματα μικρότερα του αρχικού, τα οποία συνθέτουν έναν «οριακό» γράφο (boundary graph). Μέσω αυτής της τεχνικής, επιτυγχάνεται η μείωση του *I/O* κόστους του αλγόριθμου καθώς και η μείωση των απαιτούμενων buffer μνήμης (προσωρινή αποθήκευση στοιχείων) κατά τη διεξαγωγή ερωτημάτων. Ο ιεραρχικός γράφος αποτελεί μια δύο επιπέδων αναπαράσταση του αρχικού γράφου. Το κατώτερο επίπεδο (lower level) αποτελείται από μικρά τμήματα του αρχικού γράφου, ενώ το ανώτερο επίπεδο από οριακούς κόμβους (boundary nodes) οι οποίοι ανήκουν στον «οριακό» γράφο (boundary graph). Οι οριακοί κόμβοι, ορίζονται ως ένα σύνολο κόμβων που έχουν ένα γείτονα κόμβο σε περισσότερα του ενός τμήματα στα οποία διασπάστηκε ο αρχικός γράφος. Οι ακμές του «οριακού» γράφου καλούνται οριακές ακμές και οι οριακοί κόμβοι σχηματίζουν ένα ενιαίο σύνολο στον οριακό γράφο, καθώς είναι πλήρως συνδεδεμένοι μεταξύ τους. Το κόστος μιας οριακής ακμής, είναι το κόστος του συντομότερου μονοπατιού μεταξύ των αντίστοιχων οριακών κόμβων, ενώ ένα οριακό μονοπάτι ορίζεται ως το συντομότερο μονοπάτι που υφίσταται εντός του οριακού γράφου [20].

Ο ιεραρχικός αλγόριθμος αναζήτησης συντομότερου μονοπατιού αποτελείται από τρία βασικά επιμέρους βήματα, τα οποία συνίστανται στην εύρεση του κατάλληλου ζεύγους οριακών κόμβων στον οριακό γράφο, στον υπολογισμό του οριακού μονοπατιού και στην επέκταση του οριακού μονοπατιού. Αρχικά, απαιτείται ο προσδιορισμός του οριακού κόμβου μέσω του οποίου το συντομότερο μονοπάτι αφήνει το τμήμα του γράφου στο οποίο ανήκει ο κόμβος αφετηρίας και εισέρχεται στο τμήμα του γράφου στο οποίο βρίσκεται ο κόμβος προορισμού. Στην περίπτωση που ο κόμβος αφετηρίας και ο κόμβος προορισμού είναι οριακοί κόμβοι, η επίλυση του προβλήματος είναι άμεση και δεν απαιτείται περαιτέρω διερεύνηση του γράφου. Εάν ο κόμβος αφετηρίας είναι εσωτερικός κόμβος και ο κόμβος προορισμού οριακός κόμβος, υπολογίζεται αρχικά το κόστος μετάβασης από τον κόμβο αφετηρίας προς το σύνολο των οριακών κόμβων του συγκεκριμένου τμήματος του γράφου στο οποίο ανήκει ο κόμβος αφετηρίας και στη συνέχεια ακολουθεί ο υπολογισμός του κόστους του συντομότερου μονοπατιού από τους οριακούς κόμβους προς τον κόμβο προορισμού. Το συντομότερο μονοπάτι μεταξύ του κόμβου αφετηρίας, του οριακού κόμβου και του κόμβου προορισμού είναι το μονοπάτι με το μικρότερο κόστος. Η ίδια διαδικασία, ακολουθείται και στην περίπτωση που ο κόμβος αφετηρίας είναι ένας οριακός κόμβος ενώ ο κόμβος προορισμού είναι εσωτερικός κόμβος, με αναστροφή του κόμβου αφετηρίας με τον κόμβο προορισμού. Τέλος, στην περίπτωση που οι κόμβοι αφετηρίας και προορισμού είναι εσωτερικοί κόμβοι, αναζητώνται αρχικά τα κόστη μετάβασης από τους εσωτερικούς στους οριακούς κόμβους και στα δύο τμήματα του γράφου στα οποία ανήκουν οι κόμβοι αφετηρίας και προορισμού. Στη συνέχεια, υπολογίζεται στον οριακό γράφο το συντομότερο μονοπάτι μεταξύ του συνόλου των οριακών κόμβων που έχουν προκύψει από την προηγούμενη αναζήτηση στα δύο επιμέρους τμήματα. Αφού βρεθεί το συντομότερο μονοπάτι στον οριακό γράφο, ο αλγόριθμος συνεχίζει και ολοκληρώνεται με την επέκταση του συντομότερου μονοπατιού, εκτελώντας ερωτήματα εύρεσης συντομότερης διαδρομής εντός των τμημάτων του γράφου από τα οποία διέρχεται το μονοπάτι [20].

2.8.2 Τεχνικές μείωσης κόστους I/O

Καθοριστικός παράγοντας της απόδοσης μιας διαδικασίας αναζήτησης η οποία υλοποιείται με την εφαρμογή ενός αλγόριθμου, είναι καταρχάς η ορθή επιλογή του αλγόριθμου εκείνου ο οποίος είναι ο πλέον κατάλληλος και ανταποκρίνεται με το βέλτιστο δυνατό τρόπο στις ιδιαίτερες απαιτήσεις της εκάστοτε εφαρμογής. Σημαντικό ρόλο στην ταχεία ολοκλήρωση του αλγόριθμου παίζει και ο τρόπος με τον οποίο μια

ακμή είναι αποθηκευμένη στο δίσκο. Εάν τα δύο σημεία που ορίζουν την ακμή βρίσκοναι καταχωρημένα στην ίδια σελίδα δίσκου (unsplit edge), τότε το *I/O* κόστος μειώνεται, στην αντίθετη περίπτωση το *I/O* κόστος αυξάνεται. Συνεπώς, επιδιώκεται η καλύτερη δυνατή κατανομή των κόμβων στις σελίδες του δίσκου, έτσι ώστε να περιορίζεται κατά το δυνατό περισσότερο το πλήθος των ακμών τα άκρα των οποίων βρίσκονται σε διαφορετικές σελίδες στο δίσκο (split edges). Ένας δείκτης που χρησιμοποιείται για την εκτίμηση του κόστους *I/O* είναι ο δείκτης *CRR* ο οποίος υπολογίζεται από τον τύπο *CRR = Total number of unsplit edges/ Total number of edges*. Ο δείκτης αυτός καλείται *Λόγος Εναπομένουσας Συνεκτικότητας* και όσο αυξάνεται η τιμή του τόσο μειώνεται το *I/O* κόστος.

Μια δεύτερη μέθοδος που προτείνεται για τη μείωση του κόστους *I/O*, αφορά στην κατανομή των κόμβων ενός σταθμισμένου γράφου σε υποσύνολα με τρόπο τέτοιο ώστε να ελαχιστοποιείται το πλήθος των ακμών τα άκρα των οποίων ανήκουν σε διαφορετικά υποσύνολα κόμβων. Η μέθοδος *Connectivity Clustered Access Method (CCAM)*, ταξινομεί τους κόμβους του γράφου σε ομάδες μέσω του διαμελισμού του γράφου ενώ χρησιμοποιεί ένα δευτερεύον ευρετήριο προκειμένου να υποστηριχθούν οι λειτουργίες *Find()*, *get-a- Successor()* και *get- Successors()* της κλάσης *Graph*. Το δευτερεύον ευρετήριο που χρησιμοποιείται κατά περίπτωση εξαρτάται από τις απαιτήσεις της εκάστοτε εφαρμογής. Για κάθε κόμβο, αποθηκεύονται τα στοιχεία του (συντεταγμένες, προκάτοχοι και διάδοχοι κόμβοι). Μία λίστα προκατόχων (predecessor list), περιλαμβάνει το σύνολο των ακμών που καταλήγουν στον κόμβο (incoming edges), ενώ μια λίστα διαδόχων (successor list) περιλαμβάνει το σύνολο των ακμών που φεύγουν από τον κόμβο (outgoing edges). Κάθε ακμή αναπαρίσταται από τους κωδικούς (*ids*) των κόμβων που την ορίζουν, το *id* του κόμβου αρχής και το *id* του κόμβου τέλους και το κόστος της. Η λίστα διαδόχων καλείται επίσης λίστα γειτνίασης και χρησιμοποιείται για διάφορους υπολογισμούς στο δίκτυο. Η λίστα προκατόχων, χρησιμοποιείται για την ενημέρωση της λίστας διαδόχων κατά την υλοποίηση των λειτουργιών *Insert()* και *Delete()*. Η εφαρμογή της μεθόδου *CCAM*, συνίσταται ουσιαστικά στην κατανομή των κόμβων στις σελίδες δεδομένων με βάση τη διαμέριση του γράφου, με τελικό στόχο την ελαχιστοποίηση του λόγου *CRR*. Κάθε σελίδα δεδομένων, πρέπει να διατηρείται κατά το ήμισυ τουλάχιστο, γεμάτη. Οι εγγραφές των δεδομένων δεν είναι διατεταγμένες σύμφωνα με τον κωδικό τους, ενώ η δημιουργία πρωτεύοντος ευρετηρίου προϋποθέτει τη μετονομασία των κόμβων προκειμένου να κωδικοποιηθούν οι πληροφορίες που αναφέρονται στη σελίδα του δίσκου στην οποία βρίσκεται το *id* του κόμβου. Για το λόγο αυτό, δημιουργείται ένα δευτερεύον ευρετήριο το οποίο επιπρόσθετα υποστηρίζει την

εκτέλεση ερωτημάτων σημείου και περιοχής. Εάν το δίκτυο αφορά γεωγραφικό χώρο, για κάθε κόμβο αποθηκεύονται οι συντεταγμένες του *(x, y)*. Το χωρικό σχήμα δεικτοδότησης που αφορά τις συντεταγμένες είναι δυνατό να χρησιμοποιηθεί ως δευτερεύον ευρετήριο [20].

ΚΕΦΑΛΑΙΟ 3: ΧΩΡΙΚΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό, αναπτύσσεται ένας γενικότερος προβληματισμός που αφορά στη συλλογή, στη μοντελοποίηση και στη διαχείριση χωρικών δεδομένων καθώς και στη συμβολή των Συστημάτων Γεωγραφικών Πληροφοριών (ΣΓΠ) και των Χωρικών Βάσεων Δεδομένων (ΧΒΔ) στις παραπάνω χωρικές διαδικασίες. Τόσο τα Συστήματα Γεωγραφικών Πληροφοριών όσο και τα Συστήματα Διαχείρισης Χωρικών Βάσεων Δεδομένων (ΣΔΧΒΔ) συνιστούν τεχνολογίες αιχμής τις τελευταίες δεκαετίες, όπου η συλλογή μεγάλου όγκου χωρικών δεδομένων και η επιτακτική ανάγκη διαχείρισής τους για την υλοποίηση πάσης φύσεως εφαρμογών οδήγησαν στην ανάπτυξη των τεχνολογιών αυτών. Η αύξηση των απαιτήσεων για την αποδοτικότερη διαχείριση των χωρικών δεδομένων έχει ως αποτέλεσμα, την προώθηση της έρευνας στο συγκεκριμένο πεδίο για τη βελτίωση καθαυτών των συστημάτων, των λειτουργιών που υποστηρίζουν και των εργαλείων που χρησιμοποιούν. Η έρευνα, εστιάζει κατά κύριο λόγο στην αύξηση της ταχύτητας επεξεργασίας των χωρικών δεδομένων, στην επέκταση των δυνατοτήτων που τα συστήματα αυτά προσφέρουν στο χρήστη, στη δημιουργία φιλικού προς το χρήστη περιβάλλοντος και στη συνακόλουθη μείωση του κόστους ούτως ώστε κάθε απλός χρήστης να έχει πρόσβαση στις τεχνολογίες αυτές.

Στις παραγράφους που ακολουθούν παρουσιάζονται τα βασικά χαρακτηριστικά των Συστημάτων Διαχείρισης Χωρικών Βάσεων Δεδομένων, οι διαδικασίες σχεδιασμού μιας χωρικής βάσης δεδομένων, τα χωρικά μοντέλα που εφαρμόζονται για την αναπαράσταση των χωρικών δεδομένων, οι κυριότερες λειτουργίες μιας χωρικής βάσης δεδομένων καθώς και ορισμένα στοιχεία που αφορούν στη γλώσσα διαχείρισης των χωρικών δεδομένων μέσω της οποίας καθίσταται εφικτή η αλληλεπίδραση του χρήστη με τη χωρική βάση δεδομένων.

3.1 Το Ζήτημα της Διαχείρισης Χωρικών Δεδομένων

Η εξέλιξη επιστημονικών πεδίων το αντικείμενο των οποίων είτε άμεσα είτε έμμεσα σχετίζεται με τη μελέτη και διαχείριση χωρικών δεδομένων για την εξαγωγή πληροφορίας, οδήγησε στην ανάπτυξη πληροφοριακών συστημάτων και λογισμικού για την υποστήριξη των διαδικασιών συλλογής, αποθήκευσης, συντήρησης, ενημέρωσης, διαχείρισης και ανάκτησης των χωρικών δεδομένων. Με την έννοια «χωρικά δεδομένα», νοείται μια ευρεία κατηγορία δεδομένων τα οποία είτε άμεσα είτε έμμεσα χαρακτηρίζονται από μια συγκεκριμένη θέση στο χώρο η οποία με τη σειρά της ορίζεται ως προς κάποιο σύστημα αναφοράς.

Η ραγδαία ανάπτυξη της τεχνολογίας και η συνακόλουθη υιοθέτηση των καινοτομιών και των τεχνολογικών εργαλείων από επιμέρους επιστημονικά πεδία, είχε ως αποτέλεσμα τη συλλογή μεγάλου όγκου χωρικών δεδομένων τα οποία έπρεπε να καταστούν διαχειρίσιμα για περαιτέρω επεξεργασία και εξαγωγή χρήσιμης πληροφορίας. Η φύση των χωρικών δεδομένων διαφέρει κατά περίπτωση ανάλογα με τον τρόπο και τα συστήματα που χρησιμοποιούνται για τη συλλογή τους αλλά και το σκοπό για τον οποίο συλλέγονται. Συνεπώς, απαντώνται ως τηλεπισκοπικά δεδομένα (δορυφορικές εικόνες), ως φωτογραμμετρικά δεδομένα (αεροφωτογραφίες), ως χαρτογραφικά δεδομένα (χάρτες), ως γεωδαιτικά δεδομένα (απευθείας μετρήσεις στη φυσική γήινη επιφάνεια) είτε ακόμη ως χωρικά δεδομένα που χρησιμοποιούνται από την επιστήμη της ιατρικής για τη χαρτογράφηση του ανθώπινου σώματος.

Τη διαδικασία συλλογής χωρικών δεδομένων, διαδέχονται οι διαδικασίες αποθήκευσης και επεξεργασίας τους εάν αυτό κριθεί αναγκαίο, προκειμένου να μετασχηματιστούν στην εκάστοτε επιθυμητή μορφή και να καταστούν διαχειρίσιμα από ένα πλήθος εφαρμογών. Ο τρόπος με τον οποίο οι χρήστες αλληλεπιδρούν συνήθως με τα χωρικά δεδομένα συνίσταται στη διατύπωση χωρικών ερωτημάτων (spatial queries) και την ανάκτηση της ζητούμενης πληροφορίας με την υποστήριξη ενός ΣΔΧΒΔ. Με την έννοια χωρικό ερώτημα νοείται ένα ερώτημα που αφορά χωρική πληροφορία, δηλαδή πληροφορία που σχετίζεται με το χώρο, όπως για παράδειγμα το ερώτημα «Ποιό είναι το κοντινότερο φαρμακείο από το σημείο στο οποίο βρίσκεται το σπίτι του κατοίκου μιας πόλης» ή «Ποιά είναι η απόσταση μεταξύ δύο κτιρίων» [20]. Τέτοιου είδους ερωτήματα συναντώνται πολύ συχνά σήμερα, εξυπηρετώντας χρήστες υπηρεσιών πλοήγησης όπως για παράδειγμα ενός οδηγού που μετακινείται στο οδικό δίκτυο μιας πόλης σύμφωνα με τις οδηγίες που λαμβάνει το GPS του οχήματός του, ή ο χρήστης ενός κινητού τηλεφώνου που χρησιμοποιεί την υπηρεσία πλοήγησης που διαθέτει η συσκευή του ή ο

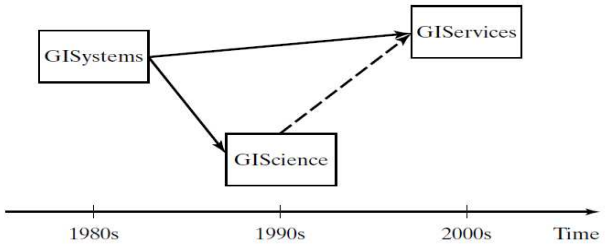
χρήστης του διαδικτύου ο οποίος αναζητά σημεία ή διαδρομές μέσω των χαρτογραφικών εφαρμογών και των υπηρεσιών πλοήγησης του WEB (Google Earth, Google Maps, Open Street Maps κ.ά.).

Η αποθήκευση, διαχείριση και ανάκτηση δεδομένων είναι διαδικασίες άμεσα συνυφασμένες με τα Συστήματα Βάσεων Δεδομένων (ΣΒΔ), τα οποία παρέχουν τη δυνατότητα τέτοιου είδους εφαρμογών. Ο τομέας των Βάσεων Δεδομένων έχει εξαπλωθεί ευρέως, εξυπηρετώντας τις ανάγκες επιστημονικών φορέων, διοικητικών οργανισμών και επιχειρήσεων οι οποίες μεταφράζονται στην ανάγκη αποθήκευσης, συντήρησης, ενημέρωσης και ανάκτησης πολύ μεγάλου όγκου δεδομένων. Ωστόσο, τα παραδοσιακά Συστήματα Βάσεων Δεδομένων κρίνονται ανεπαρκή ως προς τις δυνατότητες που παρέχουν για τη διαχείριση χωρικών δεδομένων καθώς οι απαιτήσεις διαχείρισης είναι περισσότερες και σχετικά διαφοροποιημένες σε σχέση με τα παραδοσιακά δεδομένα. Για την αναπαράσταση χωρικών δεδομένων, απαιτούνται ειδικοί τύποι δεδομένων, όπως για παράδειγμα οι τύποι που αναπαριστούν τη γεωμετρία τους καθώς και ειδικά ευρετήρια για τη δεικτοδότηση και την ταχύτερη προσπέλασή τους.

Παρά το γεγονός ότι η διαχείριση χωρικών δεδομένων παραπέμπει κυρίως σε γεωγραφικά δεδομένα που αξιοποιούνται κατά βάση από τις γεωεπιστήμες, οι Χωρικές Βάσεις Δεδομένων συνιστούν ένα εργαλείο που υιοθετείται από έναν αξιόλογο αριθμό επιστημονικών πεδίων για την υλοποίηση πλήθους εφαρμογών. Ο τομέας της κινητής τηλεφωνίας για παράδειγμα, επεκτείνοντας τις παρεχόμενες υπηρεσίες στο πεδίο της πλοήγησης, διαχειρίζεται χωρικά δεδομένα μέσω των οποίων ένας χρήστης κινητού τηλεφώνου έχει τη δυνατότητα να λαμβάνει χωρική πληροφορία σχετική με τη διαδρομή που ακολουθεί. Η υλοποίηση εφαρμογών ανάλυσης και εκτίμησης κινδύνου απαιτεί την άντληση χωρικών δεδομένων για τη δόμηση σεναρίων επικινδυνότητας, όπως για παράδειγμα ο προσδιορισμός χωρικών ζωνών (buffers) που δύνανται να επηρεαστούν από την πιθανή υπερχείλιση ενός ποταμού. Ο τομέας της αστρονομίας χρησιμοποιεί χωρικά δεδομένα για τον υπολογισμό αποστάσεων και διαφόρων άλλων μεγεθών που σχετίζονται με τα ουράνια σώματα, ενώ ο τομέας της μοριακής βιολογίας διαχειρίζεται χωρικά δεδομένα για τη χαρτογράφηση του γονιδιώματος διαφόρων οργανισμών.

3.2 Συστήματα Γεωγραφικών Πληροφοριών (ΣΓΠ) και Συστήματα Διαχείρισης Χωρικών Βάσεων Δεδομένων (ΣΔΧΒΔ)

Η τεχνολογία των Συστημάτων Γεωγραφικών Πληροφοριών (ΣΓΠ/GIS) άρχισε να αναπτύσσεται κατά τη δεκαετία του 1980 και σήμερα αποτελεί ένα από τα πλέον σημαντικά ερευνητικά πεδία των γεωεπιστημών και της επιστήμης των υπολογιστών. Πρόκειται για συστήματα που χρησιμοποιούνται για την ανάλυση και οπτικοποίηση γεωγραφικών δεδομένων, δηλαδή χωρικών δεδομένων τα οποία ορίζονται σε σχέση με ένα σύστημα αναφοράς το οποίο με τη σειρά του έχει ως επιφάνεια αναφοράς τη φυσική γήινη επιφάνεια [20]. Τα GIS εμφανίστηκαν στην αρχή ως ένα λογισμικό (GISystem), το οποίο σχεδιάστηκε για την αναπαράσταση γεωγραφικής πληροφορίας και απευθυνόταν κυρίως σε εξειδικευμένους χρήστες, στη συνέχεια εξελίχθηκε σε ένα αυτοτελές επιστημονικό πεδίο (GISience) η έρευνα του οποίου εστίαζε στην ανάπτυξη και περαιτέρω βελτίωση των Συστημάτων Γεωγραφικών Πληροφοριών μέσα από την αξιοποίηση της άλγεβρας χαρτών (map algebra) και τη μοντελοποίηση των χωρικών λειτουργιών (spatial operations), ενώ σήμερα συνιστούν πλέον συστήματα παροχής «γεωγραφικών» υπηρεσιών (GIServices) (π.χ. πλοήγηση) σε διάφορους χρήστες του παγκόσμιου ιστού και της κινητής τηλεφωνίας. Τα τρία στάδια εξέλιξης των GIS παρουσιάζονται στο σχήμα που ακολουθεί.



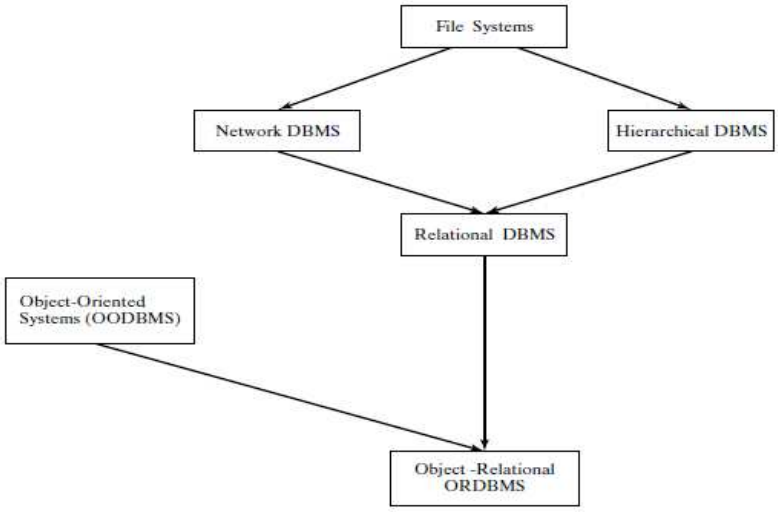
Σχήμα 3-1: Η εξέλιξη των Συστημάτων Γεωγραφικών Πληροφοριών
Πηγή: [20]

Μερικές από τις αναλυτικές διαδικασίες που ένας χρήστης δύναται να υλοποιήσει χρησιμοποιώντας ένα Σύστημα Γεωγραφικών Πληροφοριών είναι η ανάλυση αναγλύφου/ κλίσεων (terrain analysis), η ανάλυση χωρικών ροών (flow analysis), η χωρική ανάλυση δεδομένων και η εξαγωγή στατιστικών που αφορούν χωρικά δεδομένα (spatial analysis/ statistics), η υλοποίηση μετρήσεων όπως η μέτρηση αποστάσεων (measurements) κ.ά.

Η τεχνολογία των ΣΓΠ είναι στενά συνδεδεμένη με τα ΣΔΧΒΔ, καθώς το ένα σύστημα παρέχει στο άλλο σημαντικές δυνατότητες βελτίωσης και περαιτέρω επέκτασης των δυνατοτήτων τους. Τα ΣΓΠ παρέχουν στο χρήστη τη δυνατότητα υλοποίησης μιας σειράς διαδικασιών σε έναν περιορισμένο αριθμό μεμονωμένων αντικειμένων (objects) ή θεματικών επιπέδων (layers), ενώ ένα ΣΔΧΒΔ παρέχει τη δυνατότητα εκτέλεσης απλών διαδικασιών σε ένα σύνολο αντικειμένων (set of objects) ή ένα σύνολο θεματικών επιπέδων (set of layers). Με άλλα λόγια, ένα ΣΔΧΒΔ δύναται να απαντήσει σε χωρικά ερωτήματα που τίθενται σε αυτό αξιοποιώντας μαθηματική γνώση προερχόμενη από τη θεωρία συνόλων. Επιπλέον, είναι σχεδιασμένο ώστε να παρέχει τη δυνατότητα διαχείρισης πολύ μεγάλου όγκου δεδομένων τα οποία είναι αποθηκευμένα στη δευτερεύουσα μνήμη του υπολογιστή (σκληρός δίσκος κ.ά.) παρέχοντας ειδικές δομές δεικτοδότησης χωρικών δεδομένων (χωρικά ευρετήρια), χωρικούς τελεστές και χωρικές συναρτήσεις που χρησιμοποιούνται κατά τη διαδικασία ανάκτησης πληροφορίας και τη διατύπωση χωρικών ερωτημάτων. Παράλληλα, υπάρχει δυνατότητα ταυτόχρονης πρόσβασης στα χωρικά δεδομένα της βάσης από πολλούς χρήστες, όπως συμβαίνει και με τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων.

Η αποθήκευση και διαχείριση της γεωγραφικής πληροφορίας, όπως ήδη αναφέρθηκε, απαιτεί τη χρήση ειδικών τύπων αποθήκευσης και δομών δεικτοδότησης των χωρικών δεδομένων καθώς και ειδικούς χωρικούς τελεστές και συναρτήσεις για τη διαχείριση των χωρικών δεδομένων και την ανάκτηση πληροφορίας. Οι σχεσιακές βάσεις δεδομένων δεν παρέχουν αυτή τη δυνατότητα, καθώς δεν υποστηρίζουν τους ανάλογους τύπους και τις ανάλογες δομές. Ως εκ τούτου, η διαδικασία αποθήκευσης χωρικών δεδομένων σε μια σχεσιακή βάση δεδομένων είναι μια διαδικασία εξαιρετικά πολύπλοκη όπου προϋποθέτει την επανάληψη δεδομένων σε διαφορετικούς πίνακες της βάσης. Η αποθήκευση των ορίων ενός οικοδομικού τετραγώνου για παράδειγμα σε μια σχεσιακή βάση δεδομένων απαιτεί τη δημιουργία τριών πινάκων (πίνακας πολυγώνων, πίνακας πολυγραμμών, πίνακας σημείων) προκειμένου να καταστεί δυνατή η περιγραφή των πλευρών (πολυγραμμές) του οικοδομικού τετραγώνου. Η αντιμετώπιση των αδυναμιών που παρουσιάζουν οι σχεσιακές βάσεις δεδομένων ως προς τη διαχείριση χωρικών δεδομένων είναι εφικτή, μέσα από την αξιοποίηση των δυνατοτήτων του αντικειμενοστρεφούς προγραμματισμού, όπως για παράδειγμα η δυνατότητα ορισμού αφηρημένων τύπων δεδομένων (ADTs) από το χρήστη, οι οποίοι περιγράφουν τη γεωμετρία των χωρικών δεδομένων. Συνεπώς, η ενσωμάτωση ADTs, εργαλείων και μεθόδων του αντικειμενοστρεφούς προγραμματισμού στις σχεσιακές βάσεις δεδομένων, οδηγεί στη δημιουργία του αντικειμενο- σχεσιακού μοντέλου επάνω στο οποίο στηρίζεται η

τεχνολογία των αντικειμενο-σχεσιακών βάσεων δεδομένων οι οποίες παρέχουν τη δυνατότητα αποθήκευσης και διαχείρισης χωρικών δεδομένων. Στο σχήμα που ακολουθεί, παρουσιάζεται η εξέλιξη των μοντέλων που υιοθετήθηκαν διαχρονικά στο σχεδιασμό των βάσεων δεδομένων - Σύστημα Αρχείων, Ιεραρχικό Μοντέλο, Δικτυωτό Μοντέλο, Σχεσιακό Μοντέλο - έως την ανάπτυξη των Αντικειμενοσχεσιακών Συστημάτων Διαχείρισης Βάσεων Δεδομένων (ORDBMSs) [20].



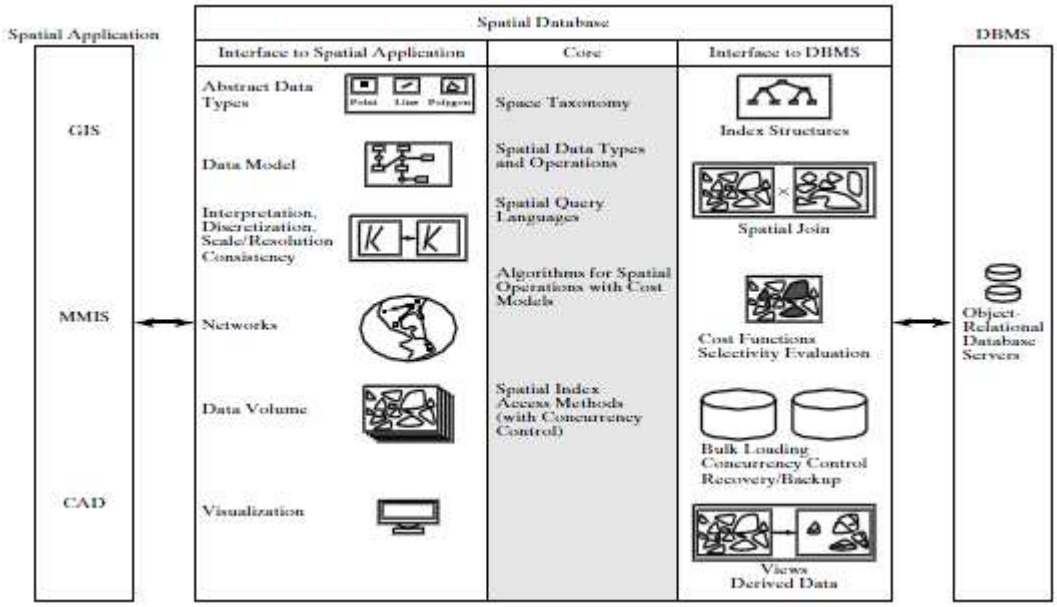
Σχήμα 3-2: Η Εξέλιξη των Συστημάτων Διαχείρισης Βάσεων Δεδομένων

Πηγή: [20]

3.3 Αρχιτεκτονική Τριών Επιπέδων

Ένα Σύστημα Διαχείρισης Χωρικών Βάσεων Δεδομένων, είναι ένα πακέτο λογισμικού που αναπτύσσεται επάνω σε ένα υποκείμενο Σύστημα Διαχείρισης Βάσεων Δεδομένων αντικειμενοστρεφές ή αντικειμενοσχεσιακό, το οποίο υποστηρίζει πολλαπλά χωρικά μοντέλα, αφηρημένους χωρικούς τύπους δεδομένων (ADTs) και μια γλώσσα διαχείρισης δεδομένων, ενώ παράλληλα διαθέτει ειδικούς τύπους δεικτοδότησης χωρικών δεδομένων (χωρικά ευρετήρια), αποδοτικούς αλγόριθμους για την υλοποίηση μιας πληθώρας χωρικών διαδικασιών καθώς και μια σειρά κανόνων για τη βελτιστοποίηση των χωρικών ερωτημάτων [20]. Η δόμηση ενός ΣΔΧΒΔ στηρίζεται στην αρχιτεκτονική

τριών επιπέδων, το πρώτο από τα οποία αφορά μια χωρική εφαρμογή, όπως για παράδειγμα ένα ΣΓΠ, το δεύτερο τη χωρική βάση δεδομένων και το τρίτο το ΣΔΧΒΔ. Το πρώτο επίπεδο αλληλεπιδρά έμμεσα με το τρίτο επίπεδο, μέσω της χωρικής βάσης δεδομένων στην οποία βρίσκεται καταχωρημένη η χωρική πληροφορία. Η αρχιτεκτονική των τριών επιπέδων, βάσει της οποίας δομείται ένα ΣΔΧΒΔ, παρουσιάζεται στο σχήμα που ακολουθεί:



Σχήμα 3-3: Αρχιτεκτονική Τριών Επιπέδων

Πηγή: [20]

3.4 Μοντελοποίηση Χωρικών Δεδομένων

Οποιαδήποτε προσπάθεια αναπαράστασης αντικειμένων ή φαινομένων του φυσικού κόσμου με τη βοήθεια ενός μοντέλου προϋποθέτει τον ακριβή ορισμό των ιδιοτήτων και των χαρακτηριστικών του, έτσι ώστε να διασφαλιστεί η υιοθέτηση εκείνου του μοντέλου που περιγράφει πλήρως και επακριβώς το προς μοντελοποίηση αντικείμενο ή φαινόμενο. Στην περίπτωση του χώρου, η διαδικασία μοντελοποίησής του είναι αρκετά σύνθετη και πολύπλοκη, γεγονός που απορρέει από τη δυσκολία απόδοσης ενός πλήρους και ενιαίου ορισμού για το «τι είναι ακριβώς ο χώρος». Επιπρόσθετα, η μοντελοποίηση του χώρου είναι μια διαδικασία απόλυτα συνυφασμένη με τον τρόπο που ο ανθρώπινος νους αντιλαμβάνεται τη χωρική πραγματικότητα, γεγονός που συνεπάγεται την αξιοποίηση

χαρακτηριστικών των χωρικών δεδομένων τα οποία γίνονται άμεσα αντιληπτά από τον ανθρώπινο εγκέφαλο και τις αισθήσεις. Τα χαρακτηριστικά αυτά αφορούν τη θέση του αντικειμένου στο χώρο, την εγγύτητά του με άλλα αντικείμενα του χώρου, τον προσανατολισμό του, το μέγεθός του, τη σύνδεσή του με άλλα αντικείμενα κ.λπ.

Στα πλαίσια της παρούσας εργασίας, το ζήτημα της μοντελοποίησης των χωρικών αντικειμένων και φαινομένων εντοπίζεται στο επίπεδο της αναπαράστασής τους, προκειμένου να καταστούν διαχειρίσιμα από τον Η/Υ. Για το σκοπό αυτό, αξιοποιούνται οι γεωμετρικές τους ιδιότητες, οι τοπολογικές σχέσεις που υφίστανται μεταξύ τους, καθώς και ένας αριθμός ιδιοτήτων που χαρακτηρίζουν τα χωρικά αντικείμενα και τα χωρικά δίκτυα. Οι τέσσερις βασικοί τρόποι που έχουν επικρατήσει για τη μοντελοποίηση του χώρου αφορούν την τοπολογική αναπαράσταση των σχέσεων που αναπτύσσονται μεταξύ των χωρικών αντικειμένων, τη δικτυακή αναπαράσταση του χώρου, την αναπαράσταση σύμφωνα με την κατεύθυνση και τον προσανατολισμό των χωρικών αντικειμένων και τέλος, την αναπαράσταση του Ευκλείδιου χώρου που προκύπτει από την αξιοποίηση των γεωμετρικών ιδιοτήτων των αντικειμένων. Το μοντέλο που υιοθετείται κάθε φορά για την αναπαράσταση του χώρου είναι αυτό, που περιγράφει με τον καλύτερο δυνατό τρόπο τα χαρακτηριστικά του εκάστοτε χωρικού προβλήματος και προσαρμόζεται πλήρως στις ανάγκες της εκάστοτε εφαρμογής. Συνεπώς, ένα μοντέλο δεδομένων ορίζεται ως ένας κανόνας ή ένα σύνολο κανόνων που εφαρμόζονται για την αναγνώριση και αναπαράσταση αντικειμένων που αναφέρονται στο χώρο [20].

Τα δύο κυριότερα μοντέλα που χρησιμοποιούνται για την αναπαράσταση των γεωγραφικών οντοτήτων και την περιγραφή των σχέσεων που αναπτύσσονται μεταξύ τους είναι το μοντέλο πεδίου (field model) και το μοντέλο οντοτήτων ή μοντέλο αντικειμένων (object model).

Το μοντέλο οντοτήτων είναι το πλέον κατάλληλο, για την περιγραφή διακριτών οντοτήτων που βρίσκονται σε συγκεκριμένη θέση στο χώρο και χαρακτηρίζονται από τις διαστάσεις τους, το σχήμα τους, τα περιγραφικά χαρακτηριστικά τους και τις σχέσεις που αναπτύσσουν με τις υπόλοιπες οντότητες που υφίστανται στο χώρο- τοπολογικές σχέσεις, σχέσεις εγγύτητας, σχέσεις απόστασης κ.λπ. Το μοντέλο οντοτήτων είναι εννοιολογικό και υλοποιείται στον υπολογιστή με εφαρμογή της διανυσματικής δομής δεδομένων (vector data structure), όπου τα αντικείμενα μοντελοποιούνται ως σημεία, γραμμές και πολύγωνα ή ως σύνθετες γεωμετρικές οντότητες οι οποίες προκύπτουν από το συνδυασμό των παραπάνω απλών γεωμετριών.

Το μοντέλο πεδίου χρησιμοποιείται για την αναπαράσταση συνεχών φαινομένων του γεωγραφικού χώρου, οι διαστάσεις και τα όρια των οποίων δεν είναι σαφώς

προσδιορισμένα, όπως για παράδειγμα η θερμοκρασία. Το πεδίο περιγράφεται με τη βοήθεια μιας συνάρτησης οι τιμές τις οποίας μεταβάλλονται για κάθε διαφορετική θέση του χώρου. Οι βασικές λειτουργίες που λαμβάνουν χώρα στο μοντέλο πεδίου, διακρίνονται στις τοπικές/ σημειακές (local), τις εστιακές (focal) και τις λειτουργίες ζώνης (zonal). Το μοντέλο πεδίου υλοποιείται στον υπολογιστή με εφαρμογή της ψηφιδωτής δομής δεδομένων (raster data structure), όπου με τη βοήθεια ενός κανάβου ο συνεχής χώρος διακρίνεται σε επιμέρους χωρικά τμήματα, τα φατνία. Η τιμή που λαμβάνει κάθε φατνίο προκύπτει ως ο μέσος όρος των τιμών των σημείων που περιλαμβάνονται στο συγκεκριμένο φατνίο καθώς οι τιμές του πεδίου είναι χωρικά αυτοσυσχετιζόμενες. Άλλες δημοφιλείς δομές που υλοποιούν το μοντέλο πεδίου στον Η/Υ, είναι η δημιουργία δικτύου ακανόνιστων τριγώνων TIN (Triangulated Irregular Network), η αναπαράσταση του χωρικού ανάγλυφου με τη βοήθεια ισοϋψών καμπυλών και το πλέγμα σημείων.

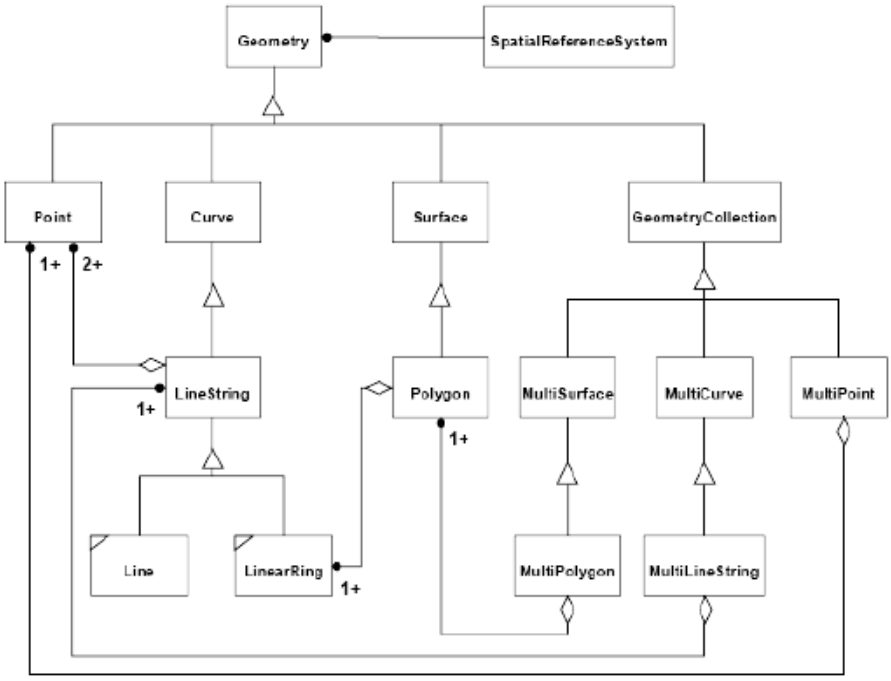
3.5 Τύποι Χωρικών Δεδομένων

Η διαχείριση χωρικών δεδομένων απαιτεί την επέκταση των δυνατοτήτων των παραδοσιακών ΣΔΒΔ, καθώς τα συστήματα αυτά σχεδιάστηκαν προκειμένου να διαχειρίζονται απλούς τύπους δεδομένων όπως αλφαριθμητικά, ακεραίους κ.λπ. Ωστόσο, τα χωρικά δεδομένα δεν είναι δυνατό να περιγραφούν με τη χρήση απλών τύπων δεδομένων. Απαιτείται η δυνατότητα ορισμού νέων τύπων δεδομένων, οι οποίοι θα ενσωματώνουν τα χαρακτηριστικά που σχετίζονται κυρίως με τη γεωμετρία των χωρικών δεδομένων. Παράλληλα, σημαντικό είναι και το ζήτημα της δεικτοδότησης των χωρικών δεδομένων ώστε να καθίσταται αποτελεσματική και ταχύτερη η διαδικασία τρεξίματος χωρικών ερωτημάτων.

Προκειμένου να αρθούν οι παραπάνω περιορισμοί, οι κατασκευαστές λογισμικού πρότειναν την επέκταση της SQL2 με πρόσθετους τύπους δεδομένων, προκειμένου να καταστεί δυνατή η διαχείριση χωρικών δεδομένων. Η γλώσσα SQL3 συνιστά την επέκταση της SQL2 και ενσωματώνει αντικειμενοστρεφείς έννοιες, μέσω των οποίων καθίσταται δυνατή η περιγραφή γεωγραφικών οντοτήτων. Υποστηρίζεται από τα αντικειμενοστρεφή και αντικειμενοσχεσιακά ΣΔΒΔ και επιτρέπει τον ορισμό αφηρημένων τύπων δεδομένων (ADTs) οι οποίοι βασίζονται στο μοντέλο αντικειμένων, παρέχουν τη δυνατότητα προσδιορισμού τοπολογικών σχέσεων και υποστηρίζουν λειτουργίες χωρικής ανάλυσης. Έμφαση δόθηκε επίσης και στη δυνατότητα της οπτικής

αναπαράστασης των αποτελεσμάτων που προκύπτουν από την εκτέλεση χωρικών ερωτημάτων, μέσα από τη σύνδεση των ΣΔΧΒΔ με λογισμικά που επιτρέπουν την οπτικοποίηση των χωρικών δεδομένων ArcGIS, QGIS κ.λπ.

Παράλληλα, προκειμένου να αποφεύγεται η σύγχυση κατά τη διαδικασία ορισμού χωρικών τύπων δεδομένων, γεγονός που συνεπάγεται τη δημιουργία προβλημάτων διαλειτουργικότητας μεταξύ των διαφορετικών συστημάτων, το Open Geospatial Consortium (OGC), πρότεινε ένα ενιαίο πρότυπο για την αναπαράσταση των γεωμετρικών οντοτήτων. Στο μοντέλο αυτό, υφίσταται στην κορυφή της ιεραρχίας η κλάση «Γεωμετρία» βάσει της οποίας μια χωρική οντότητα αναπαρίσταται ως αντικείμενο. Οι βασικές υποκλάσεις της κλάσης «Γεωμετρία» είναι το σημείο, η καμπύλη, η επιφάνεια και η συλλογή γεωμετρίας. Το μοντέλο συμπληρώνεται με τις υποκλάσεις που περιγράφονται στα υπόλοιπα επίπεδα της ιεραρχίας. Το πρότυπο του OGC υιοθετείται από την πλειοψηφία των σύγχρονων ΣΔΧΒΔ είτε εξ ολοκήρου είτε τμηματικά, καθώς ορισμένα συστήματα ενσωματώνουν μόνο τους βασικούς γεωμετρικούς τύπους.



Σχήμα 3-4: Πρότυπο Χωρικών Τύπων Δεδομένων κατά OGC

Πηγή: [20]

3.6 Διεξαγωγή Χωρικών Ερωτημάτων

Κατά τη διαδικασία διεξαγωγής ερωτημάτων σε μια βάση δεδομένων, ο χρήστης διατυπώνει ένα SQL ερώτημα στη βάση προκειμένου να ανακτήσει την επιθυμητή πληροφορία. Στη συνέχεια, το ΣΔΒΔ επεξεργάζεται το ερώτημα προκειμένου να ξεκινήσει η διαδικασία της ανάκτησης δεδομένων από τη βάση ακολουθώντας μια αλληλουχία βημάτων. Η επεξεργασία των δεδομένων αφορά σε διαδικασίες αναζήτησης στους πίνακες της βάσης και στις εγγραφές που βρίσκονται αποθηκευμένες σε αυτούς.

Βάσει του τρόπου αναζήτησης των δεδομένων, τα ερωτήματα διακρίνονται σε δύο βασικές κατηγορίες, τα single-scan και τα multi-scan ερωτήματα. Στην πρώτη περίπτωση, απαιτείται η αναζήτηση δεδομένων ανάμεσα στις εγγραφές ενός μόνο πίνακα. Κάθε εγγραφή, πρέπει να προσπελαστεί το πολύ μια φορά και στη χειρότερη περίπτωση, η ανάκτηση του επιθυμητού αποτελέσματος απαιτεί τον έλεγχο του συνόλου των εγγραφών του πίνακα προκειμένου αυτές να αξιολογηθούν ως προς το βαθμό που ικανοποιούν τα κριτήρια του ερωτήματος. Έτσι, ο χρόνος επεξεργασίας των δεδομένων περιορίζεται σε μεγάλο βαθμό εάν η πρώτη εγγραφή είναι η απάντηση στο ερώτημα που θέτει ο χρήστης, ενώ αυξάνεται ανάλογα με το πλήθος των εγγραφών που πρέπει να ελεγχθούν. Στη δεύτερη περίπτωση, η ανάκτηση των δεδομένων απαιτεί το συνδυασμό δύο ή περισσοτέρων πινάκων της βάσης προκειμένου να εξαχθεί το τελικό αποτέλεσμα.

Θεμελιώδης λειτουργία κατά τη διεξαγωγή multi-scan ερωτημάτων, συνιστά αυτή της ένωσης (join) δύο ή περισσοτέρων πινάκων. Οι πίνακες συνδέονται βάσει ενός κοινού πεδίου το οποίο περιλαμβάνει δεδομένα του ιδίου τύπου. Στην περίπτωση διεξαγωγής multi-scan χωρικών ερωτημάτων, η διαδικασία της ένωσης πινάκων αναφέρεται ως χωρική σύνδεση πινάκων (spatial join). Η χωρική σύνδεση πινάκων απαιτεί την ύπαρξη γεωμετρικών πεδίων στους πίνακες που πρόκειται να συνδεθούν, στα οποία όμως είναι δυνατό να φιλοξενούνται γεωμετρικά χαρακτηριστικά διαφορετικού τύπου, π.χ. σημείο στον ένα πίνακα και πολύγωνο στον άλλο. Όταν ένα ερώτημα σύνδεσης πινάκων απαιτεί τη σύνδεση περισσοτέρων των δύο πινάκων, τότε οι πίνακες επεξεργάζονται ανά δύο [20].

Προκειμένου να ελαχιστοποιθεί το απαιτούμενο για τη διεξαγωγή χωρικών ερωτημάτων περιοχής υπολογιστικό κόστος, εφαρμόζεται από τα ΣΔΧΒΔ μια τεχνική filter – refine η οποία ολοκληρώνεται μέσα από δύο επιμέρους βήματα. Το πρώτο βήμα, αφορά σε μια διαδικασία φιλτραρίσματος και μείωσης του πλήθους των δεδομένων για τα οποία απαιτείται λεπτομερής έλεγχος της γεωμετρίας τους. Χρησιμοποιείται μια προσεγγιστική γεωμετρία μέσα στην οποία περικλείεται η εκάστοτε γεωμετρική

οντότητα, η οποία είναι ουσιαστικά το ελάχιστο περιγεγραμμένο ορθογώνιο (Minimum Bounding Rectangle – MBR). Στο δεύτερο βήμα, χρησιμοποιούνται τα αποτελέσματα που προέκυψαν από το στάδιο του φιλτραρίσματος, προκειμένου να ελεγχθούν λεπτομερώς οι γεωμετρίες των οντοτήτων που προέκυψαν και να εξαχθεί το τελικό αποτέλεσμα [20].

3.7 Δεικτοδότηση Χωρικών Δεδομένων

Πέραν από τις επιμέρους τεχνικές που εφαρμόζονται για τη βελτιστοποίηση του υπολογιστικού κόστους κατά τη διαδικασία διεξαγωγής και επεξεργασίας χωρικών ερωτημάτων, σημαντικό ρόλο στο χρόνο ανάκτησης δεδομένων και στο αποδοτικό τρέξιμο των αλγόριθμων βάσει των οποίων υλοποιείται η επεξεργασία των δεδομένων, παίζει ο καλός φυσικός σχεδιασμός της βάσης.

Οι βάσεις δεδομένων συνιστούν μια τεχνολογία η οποία σχεδιάστηκε, προκειμένου να καθίσταται δυνατή η αποθήκευση μεγάλου όγκου δεδομένων. Ως εκ τούτου, η αποθήκευση των δεδομένων στην κύρια μνήμη του υπολογιστή είναι αδύνατη και συνεπώς είναι απαραίτητη η αποθήκευση δεδομένων στη δευτερεύουσα μνήμη. Το πρόβλημα που ανακύπτει στην περίπτωση αυτή συνίσταται στο γεγονός ότι, ο χρόνος προσπέλασης των δεδομένων που βρίσκονται αποθηκευμένα στη δευτερεύουσα μνήμη είναι βραδύτερος σε σχέση με το χρόνο που απαιτείται για την προσπέλαση δεδομένων που βρίσκονται αποθηκευμένα στην κύρια μνήμη. Προκειμένου να αρθεί ο παραπάνω περιορισμός, δίνεται έμφαση στον καλό φυσικό σχεδιασμό της βάσης, έτσι ώστε να ελαχιστοποιείται ο χρόνος εισόδου/ εξόδου των δεδομένων προς και από την κύρια μνήμη του υπολογιστή (*I/O time*).

Η διαδικασία φυσικού σχεδιασμού και δεικτοδότησης μιας χωρικής βάσης δεδομένων εστιάζει στην ελαχιστοποίηση του αριθμού των σελίδων του δίσκου, στις οποίες γίνεται κάθε φορά η προσπέλαση και η διαχείριση των δεδομένων. Τα χωρικά ευρετήρια εξασφαλίζουν την ταχύτερη ανάκτηση δεδομένων, στη βάση των σχέσεων εγγύτητας που αναπτύσσονται μεταξύ των χωρικών δεδομένων. Επομένως σε κάθε γεωμετρικό πεδίο μιας σχέσης που περιλαμβάνει δεδομένα χωρικού τύπου, πρέπει να ανατίθεται ένα χωρικό ευρετήριο προκειμένου τα δεδομένα του πεδίου να επεξεργάζονται ταχύτερα. Τα χωρικά ευρετήρια υποστηρίζουν χωρικές λειτουργίες, όπως η εύρεση των γειτονικών οντοτήτων μιας συγκεκριμένης γεωγραφικής οντότητας, η εύρεση γεωγραφικών αντικειμένων που βρίσκονται εντός των ορίων μιας πολυγωνικής περιοχής κ.λπ. Μέσα από τη διαδικασία δεικτοδότησης των χωρικών δεδομένων, επιδιώκεται η ταχύτερη

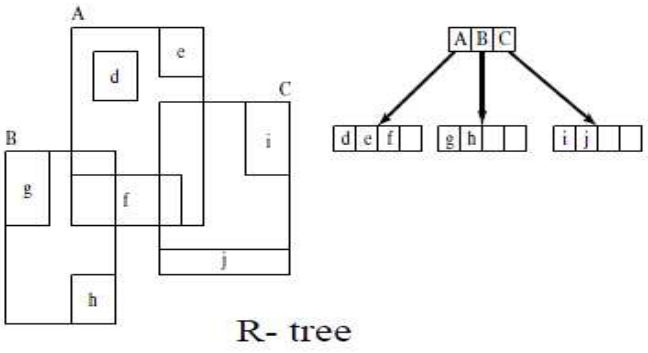
διεξαγωγή ερωτημάτων επιλογής και σύνδεσης. Η δεικτοδότηση χωρικών δεδομένων διαφέρει από τις κλασικές μεθόδους δεικτοδότησης, καθώς τα χωρικά δεδομένα συνιστούν πολυδιάστατες γεωγραφικές οντότητες και συνοδεύονται από τις συντεταγμένες τους. Συνεπώς, η αναζήτηση στην περίπτωση αυτή σχετίζεται όχι μόνο με τα περιγραφικά χαρακτηριστικά τους αλλά και με τις χωρικές τους ιδιότητες [URL2].

Η αποτελεσματική προσπέλαση και αναζήτηση χωρικών δεδομένων αφορά αφενός μεν την εξοικονόμηση μνήμης, αφετέρου δε την ταχεία ανάκτηση των δεδομένων που αναζητώνται. Οι κυριότεροι τύποι αναζητήσεων που σχετίζονται με χωρικά δεδομένα, είναι η αναζήτηση σημείου και η αναζήτηση περιοχής. Οι αναζητήσεις του παραπάνω τύπου απαιτούν την εφαρμογή χωρικών ευρετηρίων, προκειμένου να καταστεί δυνατή η προσπέλαση των χωρικών δεδομένων που βρίσκονται αποθηκευμένα στη βάση. Τα χωρικά ευρετήρια συνιστούν επεκτάσεις των απλών μονοδιάστατων ευρετηρίων και συνεισφέρουν σημαντικά στη μείωση του απαιτούμενου χρόνου προσπέλασης των χωρικών δεδομένων.

Λόγω του γεγονότος ότι ο δίσκος είναι μια μονοδιάστατη συσκευή αποθήκευσης δεδομένων, έχουν προταθεί διάφορες μέθοδοι προκειμένου να καταστεί δυνατή η αποθήκευση των πολυδιάστατων χωρικών δεδομένων στο συγκεκριμένο αποθηκευτικό μέσο μέσα από την αντιστοίχιση του ν-διάστατου χώρου, κυρίως του δυδιάστατου και του τρισδιάστατου, με το μονοδιάστατο. Οι μέθοδοι αυτού του τύπου ορίζονται ως πολυδιάστατες μέθοδοι προσπέλασης, ο σχεδιασμός τους βασίζεται σε μονοδιάστατες μεθόδους προσπέλασης όπως ο κατακερματισμός και η δενδρική δομή και διακρίνονται σε δύο βασικές κατηγορίες. Η πρώτη κατηγορία αφορά τις μεθόδους προσπέλασης περιοχών (spatial access methods), οι οποίες εφαρμόζονται για την οργάνωση δεδομένων που έχουν κάποια έκταση, όπως οι πολυγωνικές και οι γραμμικές οντότητες, ενώ η δεύτερη κατηγορία αφορά τις μεθόδους προσπέλασης σημείων (point access methods), οι οποίες εφαρμόζονται για την οργάνωση πολυδιάστατων σημείων. Στην πρώτη κατηγορία μεθόδων ανήκουν το R- δένδρο και οι παραλλαγές του, το τετραδικό δένδρο, το κυτταρικό δένδρο (cell tree), το BV- δένδρο κ.ά. Στη δεύτερη κατηγορία, ανήκουν το αρχείο ισοσταθμισμένου και φωλιασμένου πλέγματος (BANG file), το K-D-B δένδρο κ.ά.

Η συνηθέστερη δομή που χρησιμοποιείται για τη δεικτοδότηση χωρικών δεδομένων είναι το R- δένδρο, το οποίο συνιστά ένα από τα πρώτα ευρετήρια που χρησιμοποιήθηκαν για την οργάνωση πολυδιάστατων γεωγραφικών οντοτήτων. Η συγκεκριμένη δομή χρησιμοποιεί ιεραρχημένα ορθογώνια, προκειμένου να ομαδοποιήσει το χώρο και τα χωρικά αντικείμενα ορίζονται από ένα ελάχιστο περιγεγραμμένο ορθογώνιο (MBR).

Κάθε ορθογώνιο αποθηκεύεται στα φύλλα του δένδρου, ενώ στο ευρετήριο αποθηκεύονται οι δείκτες των ορθογωνίων καθώς και οι συντεταγμένες των κορυφών που τα ορίζουν. Κάθε διαδικασία αναζήτησης στο R- δένδρο ξεκινά από τη ρίζα του και επαναλαμβάνεται αναδρομικά έως την εύρεση του επιθυμητού αποτελέσματος. Το R-δένδρο, ως δομή, πλεονεκτεί τόσο ως προς τον απαιτούμενο χώρο μνήμης όσο και ως προς τις χρονικές επιδόσεις. Μειονεκτεί όμως, ως προς τις δυνατότητες που παρέχει για τη γρήγορη ενημέρωση των δεδομένων [8], [9], [15], [17].



Σχήμα 3-5: Το R- δένδρο

Πηγή: [20]

3.8 Χωρικές Λειτουργίες

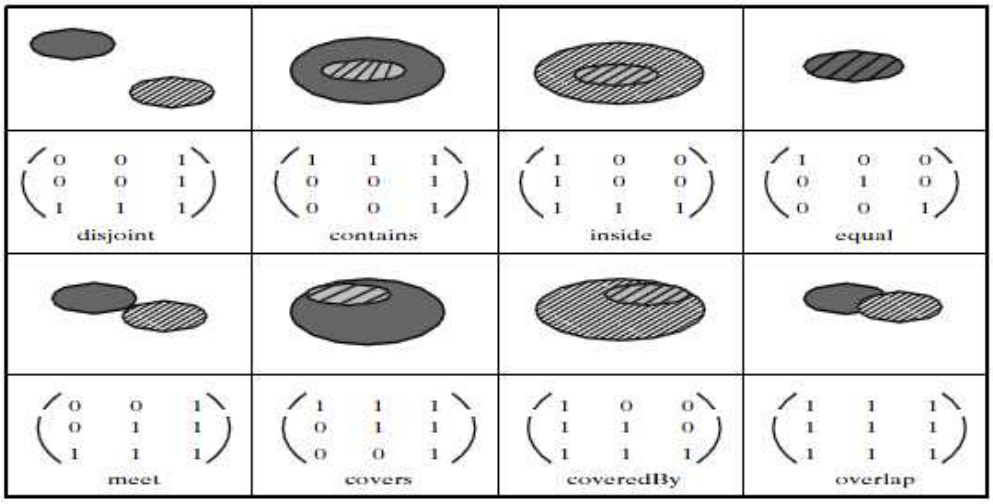
Οι χωρικές λειτουργίες αφορούν στην αλληλεπίδραση των χωρικών δεδομένων μεταξύ τους, η οποία προσδιορίζεται μέσα από την εφαρμογή χωρικών τελεστών. Οι χωρικοί τελεστές επιτρέπουν τη διεξαγωγή πράξεων μεταξύ των χωρικών δεδομένων όπως είναι η αναζήτηση τομών, επικαλύψεων κ.λπ., προσδιορίζοντας με τον τρόπο αυτό διάφορες ιδιότητες και σχέσεις που υφίστανται μεταξύ των γεωγραφικών οντοτήτων.

Οι χωρικοί τελεστές, ανάλογα με τη λειτουργία που υλοποιούν επάνω στα χωρικά δεδομένα, κατηγοριοποιούνται σε επιμέρους ομάδες. Αρκετές ταξινομήσεις των χωρικών τελεστών έχουν προταθεί από διάφορους ερευνητές αλλά και από το OGC. Μια βασική ομαδοποίηση των χωρικών τελεστών είναι αυτή που τους διακρίνει σε τοπολογικούς και μη. Στους τοπολογικούς τελεστές περιλαμβάνονται λειτουργίες που σχετίζονται με τις

τοπολογικές σχέσεις που υφίστανται μεταξύ των γεωγραφικών οντοτήτων, όπως η επικάλυψη μεταξύ δύο πολυγωνικών οντοτήτων, η γειτνίαση, η συμπερίληψη ενός σημείου από ένα πολύγωνο κ.ά. Οι τοπολογικές ιδιότητες των χωρικών οντοτήτων παραμένουν ανεπηρέαστες από οποιαδήποτε αλλαγή προκύψει κατά τη διαδικασία αναπαράστασης των χωρικών δεδομένων, καθώς υφίστανται ούτως ή άλλως και προσδιορίζουν τις σχέσεις που υφίστανται μεταξύ των χωρικών αντικειμένων. Σύμφωνα με τον Egenhofer, ο προσδιορισμός της δυαδικής τοπολογικής σχέσης που υφίσταται μεταξύ δύο χωρικών οντοτήτων *A* και *B* που κείνται σε ένα επίπεδο R^2, στηρίζεται στην τομή του εσωτερικού του *A* (A^0), του ορίου του *A* (∂A) και του εξωτερικού του *A* (A^-), με το εσωτερικό του *B* (B^0), το όριο του *B* (∂B) και το εξωτερικό του *B* (B^-). Στον πίνακα των εννέα τομών που ακολουθεί, φαίνονται οι εννέα διαφορετικές τοπολογικές σχέσεις που αναπτύσσονται μεταξύ των χωρικών οντοτήτων *A* και *B*.

$$\Gamma_9(A,B) = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Οι τιμές του πίνακα των εννέα τομών μπορεί να είναι, είτε ίσες με το μηδέν (0) είτε ίσες με το ένα (1), αναλόγως με το εάν είναι αληθής ή όχι η σχέση που κάθε φορά περιγράφεται. Οι τοπολογικές σχέσεις που είναι δυνατό να υφίστανται μεταξύ δύο οντοτήτων αφορούν τη μη σύνδεση των χωρικών οντοτήτων (disjoint), τη με κάποιο τρόπο συνάντηση των χωρικών οντοτήτων (meet), την επικάλυψη (overlap), την ισότητα (equal), τη συμπερίληψη (contains), τη σχέση «εντός» (inside), την κάλυψη (covers) και την «κάλυψη από» (covered by). Οι σχέσεις του πίνακα των εννέα τομών παρουσιάζονται και στο σχήμα που ακολουθεί το οποίο καλείται «Μοντέλο των Εννέα Τομών» (Nine Intersection Model).



Σχήμα 3-6: Το Μοντέλο των Εννέα Τομών

Πηγή: [20]

Οι μη τοπολογικές σχέσεις που αφορούν τις χωρικές οντότητες, περιλαμβάνουν την ευκλείδια απόσταση μεταξύ δύο οντοτήτων, τον προσανατολισμό, το μήκος, την περίμετρο και το εμβαδό.

Στην περίπτωση που τα χωρικά δεδομένα αναπαρίστανται με τη βοήθεια του μοντέλου πεδίων, οι χωρικές λειτουργίες που υλοποιούνται σε αυτά ομαδοποιούνται σε τρεις επιμέρους κατηγορίες, οι οποίες παρουσιάζονται στη συνέχεια [20].

- Τοπικές λειτουργίες (Local operations), όπου κάθε τιμή που υπολογίζεται για κάθε θέση στο επίπεδο προκύπτει ως συνάρτηση των δεδομένων που σχετίζονται με τη συγκεκριμένη θέση.
- Εστιακές λειτουργίες (Focal operations), όπου κάθε τιμή που λαμβάνει μια θέση στο επίπεδο εξαρτάται από τις τιμές που λαμβάνουν οι γειτονικές σε αυτή θέσεις.
- Λειτουργίες ζώνης (Zonal operations), όπου η τιμή κάθε θέσης στο επίπεδο υπολογίζεται ως συνάρτηση των τιμών των θέσεων που ανήκουν στην ίδια ζώνη με την αρχική.

Τέλος, το OGC χρησιμοποιεί μια διαφοροποιημένη ταξινόμηση των χωρικών τελεστών διακρίνοντάς τους σε βασικούς τελεστές (basic operators), τοπολογικούς τελεστές (topological set operators) και τελεστές χωρικής ανάλυσης (spatial analysis). Οι

τελεστές εφαρμόζονται στο σύνολο των αντίστοιχων τύπων χωρικών δεδομένων που προτείνει το OGC και αφορούν διανυσματικούς τύπους δεδομένων.

ΚΑΤΗΓΟΡΙΑ ΤΕΛΕΣΤΗ	ΤΕΛΕΣΤΗΣ	ΧΩΡΙΚΗ ΛΕΙΤΟΥΡΓΙΑ
Βασικοί τελεστές	SpatialReference()	Επιστρέφει το σύστημα αναφοράς της γεωμετρίας
	Envelope()	Επιστρέφει το ελάχιστο περιγεγραμμένο ορθογώνιο μιας γεωμετρίας
	Export()	Επιστρέφει μια διαφορετική αναπαράσταση της γεωμετρίας
	IsEmpty()	Επιστρέφει την τιμή TRUE εάν η γεωμετρία είναι κενή
	IsSimlpe()	Επιστρέφει την τιμή TRUE εάν η γεωμετρία είναι απλή
	Boundary()	Επιστρέφει το περίγραμμα της γεωμετρίας
Τοπολογικοί τελεστές	Equal	Επιστρέφει την τιμή TRUE εάν το εσωτερικό και το περίγραμμα δύο γεωμετριών είναι μεταξύ τους ίσα
	Disjoint	Επιστρέφει την τιμή TRUE εάν το εσωτερικό και το περίγραμμα δύο γεωμετριών δεν τέμνονται μεταξύ τους
	Intersect	Επιστρέφει την τιμή TRUE εάν δύο γεωμετρίες τέμνονται μεταξύ τους
	Touch	Επιστρέφει την τιμή TRUE εάν μόνο τα περιγράμματα δύο γεωμετριών τέμνονται μεταξύ τους
	Cross	Επιστρέφει την τιμή TRUE εάν το εσωτερικό μιας επιφάνειας τέμνεται από μια καμπύλη
	Within	Επιστρέφει την τιμή TRUE εάν μια γεωμετρία βρίσκεται μέσα σε μια άλλη γεωμετρία
	Contains	Ελέγχει εάν μια δεδομένη γεωμετρία περιέχει μια άλλη γεωμετρία
	Overlap	Επιστρέφει την τιμή TRUE εάν μια γεωμετρία επικαλύπτει μια άλλη γεωμετρία
Τελεστές χωρικής ανάλυσης	Distance	Επιστρέφει τη μικρότερη απόσταση μεταξύ δύο γεωμετρικών οντοτήτων
	Buffer	Επιστρέφει μια γεωμετρία που περιέχει όλα τα σημεία η απόσταση των οποίων από μια δοσμένη γεωμετρία είναι μικρότερη ή ίση από μια προσδιορισμένη απόσταση
	ConvexHull	Επιστρέφει το μικρότερο κυρτό πολύγωνο που εσωκλείει μια γεωμετρία
	Intersection	Επιστρέφει την τομή δύο γεωμετριών
	Union	Επιστρέφει την ένωση δύο γεωμετριών
	Difference	Επιστρέφει τη διαφορά δύο γεωμετριών
	SymmDiff	Επιστρέφει τη συμμετρική διαφορά δύο γεωμετριών

Πίνακας 3-1: Κατηγοριοποίηση Χωρικών Λειτουργιών κατά OGC

Πηγή: [20]

3.9 Γλώσσες Ερωτημάτων σε Χωρικά Δεδομένα και Γράφους

Οι γλώσσες ερωτημάτων είναι το εργαλείο μέσω του οποίου οι χρήστες των Συστημάτων Διαχείρισης Βάσεων Δεδομένων αλληλεπιδρούν με τα δεδομένα που βρίσκονται αποθηκευμένα σε αυτά. Η πιο δημοφιλής γλώσσα επερωτήσεων που υιοθετείται από τα περισσότερα εμπορικά ΣΔΒΔ, είναι η SQL ο σχεδιασμός της οποίας βασίζεται στις αρχές της σχεσιακής άλγεβρας. Η SQL χρησιμοποιείται κατά κύριο λόγο στις βάσεις δεδομένων για τη διαχείριση των δεδομένων που βρίσκονται αποθηκευμένα σε αυτές και ως εκ τούτου περιλαμβάνει λειτουργίες δημιουργίας σχημάτων και πινάκων, ανάκτησης, δεικτοδότησης και ενημέρωσης δεδομένων και λειτουργίες ελέγχου πρόσβασης στα δεδομένα της βάσης.

Στην περίπτωση των Συστημάτων Διαχείρισης Χωρικών Βάσεων Δεδομένων, τα οποία συνιστούν επέκταση των κλασσικών Συστημάτων Διαχείρισης Βάσεων Δεδομένων, υπεισέρχεται η ανάγκη ταυτόχρονης διαχείρισης χωρικών και μη- χωρικών δεδομένων. Ως εκ τούτου, είναι προφανής η ανάγκη επέκτασης της SQL και ενσωμάτωσης σε αυτή χωρικών τύπων δεδομένων προκειμένου να καταστεί δυνατή η διαχείρισή τους από ένα ΣΔΧΒΔ με τη βοήθεια χωρικών τελεστών και χωρικών συναρτήσεων.

Η SQL (πρότυπο SQL92) ήταν αρχικά σχεδιασμένη προκειμένου να διαχειρίζεται απλούς τύπους δεδομένων όπως αλφαριθμητικά, ακέραιοι κ.λπ. Ωστόσο, η ενσωμάτωση σε αυτή του αντικειμενοστρεφούς μοντέλου, με τη ουσιαστική συμβολή του OGC, έδωσε τη δυνατότητα αναπαράστασης σύνθετων γεωμετρικών τύπων και ως εκ τούτου τη δυνατότητα διαχείρισης χωρικών δεδομένων, τα οποία μοντελοποιούνται είτε ως πολύγωνα, γραμμές ή σημεία είτε ως σύνθετες γεωμετρίες (πολυ- γραμμή, πολυ-επιφάνεια κ.ά.). Το νέο πρότυπο που προέκυψε μετά την επέκταση της SQL, προκειμένου να διαχειρίζεται χωρικούς τύπους δεδομένων, είναι η SQL3. Η SQL3 ενσωματώνει αντικειμενοστρεφείς τύπους, παρέχει τη δυνατότητα παράλληλης διαχείρισης χωρικών και μη-χωρικών δεδομένων και υποστηρίζει δυδιάστατους αφηρημένους τύπους δεδομένων (ADTs) όπως οι τύποι των χωρικών δεδομένων, οι γράφοι που χρησιμοποιούνται για τη μοντελοποίηση χωρικών δικτύων, τα μονοπάτια η εύρεση των οποίων αναζητάται σε γράφους κ.λπ. [4].

Στην περίπτωση των απλών χωρικών δεδεομένων, η βασική κλάση γεωμετρικών αντικειμένων είναι η κλάση *Geometry*, ενώ στην περίπτωση των γράφων ορίζονται τρεις θεμελιώδεις κλάσεις, η κλάση *Graph*, η κλάση *Vertex* και η κλάση *Edge*. Ο ορισμός των παραπάνω κλάσεων καθίσταται δυνατός με την υιοθέτηση του αντικειμενο- σχεσιακού

προτύπου SQL3 και για κάθε μια από αυτές, υφίσταται μια σειρά από χωρικές λειτουργίες όπως η πρόσθεση ή η διαγραφή κόμβου ή ακμής σε ένα γράφο, η εύρεση ενός γειτονικού κόμβου ενός κόμβου αναφοράς, η σήμανση ενός κόμβου τον οποίο έχει επισκεφθεί κάποιος αλγόριθμος, η εύρεση του κόμβου αρχής ή του κόμβου πέρατος μιας ακμής καθώς και το σύνολο των λειτουργιών που υλοποιούνται στα χωρικά δεδομένα όπως αυτές περιγράφηκαν στο προηγούμενο εδάφιο. Η υλοποίηση των παραπάνω λειτουργιών ήταν αδύνατη με την αποκλειστική εφαρμογή των σχέσεων της σχεσιακής άλγεβρας και για το λόγο αυτό, το νέο πρότυπο ενσωματώνει το αντικειμενοστρεφές μοντέλο. Οι χωρικές λειτουργίες που υλοποιούνται σε χωρικά αντικείμενα, προκύπτουν από τις σχέσεις εγγύτητας που υφίστανται μεταξύ των χωρικών δεδομένων, ενώ στην περίπτωση των χωρικών δικτύων, από τις σχέσεις συνδεσιμότητας που αναπτύσσονται μεταξύ των στοιχείων του γράφου που τα αναπαριστά [20].

ΚΕΦΑΛΑΙΟ 4: ΥΛΟΠΟΙΗΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Στο κεφάλαιο αυτό, παρουσιάζεται το πρώτο μέρος της εφαρμογής που υλοποιήθηκε στα πλαίσια της παρούσας εργασίας και αφορά στην αναπαράσταση του βασικού οδικού δικτύου της Αθήνας σε μια χωρική βάση δεδομένων. Απώτερος στόχος της εργασίας είναι η ανάπτυξη μιας εφαρμογής, η οποία θα παρέχει τη δυνατότητα εύρεσης και αναπαράστασης βέλτιστων διαδρομών μεταξύ δύο ή περισσοτέρων σημείων του οδικού δικτύου σε ένα διαδικτυακό χάρτη και η σύνδεση της διαδικτυακής εφαρμογής με τη βάση δεδομένων, για την άντληση και επεξεργασία της απαιτούμενης χωρικής πληροφορίας.

Στο πρώτο στάδιο της εφαρμογής, απαιτείται η αποθήκευση των γεωγραφικών και περιγραφικών δεδομένων του δικτύου σε μια χωρική βάση δεδομένων, από την οποία θα αντλείται η απαραίτητη πληροφορία που κάθε φορά αναζητεί ο χρήστης ενώ παράλληλα θα υλοποιούνται οι απαραίτητοι υπολογισμοί για την εξαγωγή του αποτελέσματος. Το ΣΔΧΒΔ που χρησιμοποιήθηκε για τη μοντελοποίηση του δικτύου σε χωρική βάση δεδομένων είναι η PostgreSQL 8.3, εμπλουτισμένη με τη χωρική επέκταση PostGIS και τις συναρτήσεις του pgRouting οι οποίες υλοποιούν διαδικασίες δρομολόγησης σε δίκτυα.

Στα εδάφια που ακολουθούν, γίνεται μια συνοπτική αναφορά στα βασικά χαρακτηριστικά της PostgreSQL, του PostGIS και του pgRouting, περιγράφονται οι διαδικασίες αναπαράστασης του οδικού δικτύου της Αθήνας στη βάση και οι συναρτήσεις που χρησιμοποιήθηκαν στα πλαίσια της εφαρμογής.

4.1 Η PostgreSQL 8.3

Η PostgreSQL συνιστά ένα ελεύθερα αναπτυσσόμενο λογισμικό διαχείρισης βάσεων δεδομένων, ανοικτού κώδικα. Αν και αρχικά σχεδιάστηκε για να τρέχει σε ανοικτού

κώδικα λειτουργικά συστήματα (Unix/ Linux), σήμερα διατίθενται εκδόσεις συμβατές με όλα τα λειτουργικά συστήματα (Windows, Mac OS X), ενώ παρέχει μια σειρά διεπαφών (APIs) οι οποίες υποστηρίζουν διάφορες γλώσσες προγραμματισμού όπως η Java, οι C/C++, η Python κ.ά. [URL13], [URL14].

Η PostgreSQL, ως αντικειμενο- σχεσιακό ΣΔΧΒΔ, διαθέτει χωρικούς τύπους δεδομένων (ADTs) για την αποθήκευση απλών γεωμετρικών οντοτήτων, μηχανισμούς δεικτοδότησης για την οργάνωση των χωρικών δεδομένων με χρήση των κατάλληλων ευρετηρίων (spatial indexing) και ένα σύνολο χωρικών τελεστών και συναρτήσεων για την υποστήριξη χωρικών ή συνδυασμένων ερωτημάτων. Επιτρέπει την αποθήκευση μόνο δυδιάστατων χωρικών οντοτήτων, ενώ χαρακτηρίζεται από έναν πρόσθετο περιορισμό που σχετίζεται με τη χωρική αναφορά των γεωγραφικών οντοτήτων καθώς, δεν παρέχει τη δυνατότητα προσδιορισμού συστήματος αναφοράς. Οι γεωμετρικές οντότητες παριστάνονται ως ακολουθίες συντεταγμένων και αποκλίνουν από το επίσημο πρότυπο του OGC. Ως εκ τούτου, δεν υποστηρίζεται η αναπαράσταση πολύπλοκων γεωμετρικών οντοτήτων όπως για παράδειγμα τα multilinestrings ενώ ορισμένες γεωμετρικές συναρτήσεις δεν έχουν ακόμη υλοποιηθεί στο περιβάλλον της PostgreSQL [26].

Οι απλοί γεωμετρικοί τύποι που υποστηρίζει η PostgreSQL είναι το σημείο, η γραμμή, το πεπερασμένο ευθύγραμμο τμήμα, το ορθογώνιο, το κλειστό μονοπάτι (όμοιο με το πολύγωνο), το ανοικτό μονοπάτι, το πολύγωνο και ο κύκλος. Σύνθετοι γεωμετρικοί τύποι δεν υποστηρίζονται από το συγκεκριμένο περιβάλλον. Όσον αφορά τη δεικτοδότηση των χωρικών δεδομένων, τα χωρικά ευρετήρια που υποστηρίζει η PostgreSQL περιλαμβάνουν το R- δένδρο, το οποίο όμως δύναται να χρησιμοποιηθεί για τη δεικτοδότηση μόνο πολυγωνικών οντοτήτων, και τα γενικευμένα δένδρα αναζήτησης GiST τα οποία χρησιμοποιούνται για τη δεικτοδότηση του συνόλου των χωρικών τύπων της βάσης δεδομένων (σημεία, γραμμές, πολύγωνα). Τα γενικευμένα δένδρα αναζήτησης παρέχουν τη δυνατότητα χειρισμού κενών γεωμετρικών οντοτήτων, τη δυνατότητα δημιουργίας ελάχιστων περιγεγραμμένων ορθογωνίων (MBRs) στις γεωμετρικές οντότητες ακόμη και αν αυτές αφορούν περίπλοκα σχήματα που ορίζονται από μεγάλο πλήθος συνταγμένων, ενώ παράλληλα είναι εφαρμόσιμα στην περίπτωση πολυδιάστατων δεδομένων. Οι χωρικές λειτουργίες που υποστηρίζει η PostgreSQL δεν είναι πλήρως συμβατές με τις προτεινόμενες από το OGC χωρικές λειτουργίες και υλοποιούνται με τη βοήθεια γεωμετρικών τελεστών απόστασης *(□□□)*, επικάλυψης *(&&)*, τομής *(?#)* κ.λπ. Τέλος, η PostgreSQL διαθέτει και μια σειρά υλοποιημένων γεωμετρικών συναρτήσεων μέσω των οποίων διευκολύνεται η διεξαγωγή χωρικών ερωτημάτων που αφορούν την

εύρεση του εμβαδού μιας περιοχής *(area(object))*, την εύρεση της διαμέτρου *(diameter(circle))* και της ακτίνας ενός κύκλου κ.ά. [26].

Συμπερασματικά, η PostgreSQL εξασφαλίζει την τήρηση συνέπειας και ασφάλειας ως προς την αποθήκευση και ενημέρωση των χωρικών δεδομένων που βρίσκονται αποθηκευμένα στη βάση και την ορθότητα κατά τις διαδικασίες διεξαγωγής των χωρικών λειτουργιών και υπολογισμών. Παρέχει ειδικούς χωρικούς τύπους δεδομένων βάσει των οποίων μοντελοποιούνται τα χωρικά δεδομένα και ειδικούς χωρικούς τελεστές και χωρικές συναρτήσεις για την υλοποίηση μιας πληθώρας χωρικών λειτουργιών. Ωστόσο, δεν υιοθετεί πλήρως τους χωρικούς τύπους και τους τελεστές που προτείνει το πρότυπο του OGC, ενώ ο χρήστης δεν έχει τη δυνατότητα ορισμού συστήματος αναφοράς για τα χωρικά του δεδομένα.

4.1.1 Το PostGIS

Το PostGIS συνιστά μια επέκταση της PostgreSQL, που παρέχει τη δυνατότητα υλοποίησης περισσότερων χωρικών λειτουργιών σε σχέση με αυτές που ήδη διαθέτει η PostgreSQL. Όπως και η PostgreSQL, το PostGIS είναι ένα λογισμικό ανοικτού κώδικα, ειδικά σχεδιασμένο για τη διαχείριση χωρικών δεδομένων. Τρέχει επάνω στην PostgreSQL και έχει σχεδιαστεί σύμφωνα με τις προδιαγραφές του OGC. Υποστηρίζει την αναπαράσταση γεωμετρικών οντοτήτων που έχουν περισσότερες των δύο διαστάσεων και πιο συγκεκριμένα, επιτρέπει την αναπαράσταση 2/διάστατων, 3/διάστατων και 4/διάστατων γεωμετριών. Παράλληλα, επιτρέπει στο χρήστη τον ορισμό συστήματος αναφοράς στα χωρικά δεδομένα που διαχειρίζεται [URL12], [URL13].

Το PostGIS διαθέτει ένα σύνολο βασικών και τοπολογικών χωρικών τελεστών όπως αυτοί ορίζονται από το OGC, καθώς επίσης και έναν αριθμό γεωμετρικών συναρτήσεων και συναρτήσεων γεωμετρικού ελέγχου που επιτρέπουν τη διεξαγωγή σύνθετων υπολογισμών και ελέγχων επάνω στα χωρικά δεδομένα. Ο χρήστης έχει τη δυνατότητα ορισμού συστήματος αναφοράς στα χωρικά δεδομένα του, καθώς το PostGIS παρέχει στη διάθεσή του ένα σύνολο συστημάτων αναφοράς, τυποποιημένων σύμφωνα με τις προδιαγραφές του OGC, τα οποία βρίσκονται αποθηκευμένα στον πίνακα *SPATIAL_REF_SYS* του PostGIS. Παράλληλα, ο χρήστης μπορεί να ορίσει ένα νέο σύστημα αναφοράς για τα δεδομένα του ή να μετασχηματίσει τις συντεταγμένες των δεδομένων του από ένα υφιστάμενο σύστημα αναφοράς σε ένα άλλο, με τρέξιμο της ανάλογης συνάρτησης που διαθέτει το PostGIS.

Κατά τη διαδικασία δημιουργίας των πινάκων της βάσης, δηλώνονται πρώτα τα περιγραφικά χαρακτηριστικά των γεωγραφικών οντοτήτων και στο τέλος, ορίζεται το πεδίο στο οποίο πρόκειται να αποθηκευτεί η γεωμετρία των χωρικών οντοτήτων με τη βοήθεια της συνάρτησης *AddGeometryColumn()*. Τη δημιουργία των σχέσεων της βάσης, διαδέχεται ο έλεγχος για την ορθότητα των γεωμετρικών στοιχείων σύμφωνα με τις προδιαγραφές του OGC. Τόσο η δημιουργία του πεδίου της γεωμετρίας όσο και η διαδικασία εισαγωγής των γεωμετρικών στοιχείων στη στήλη αυτή, απαιτεί ιδιαίτερη προσοχή ως προς τη σύνταξη των αντίστοιχων εντολών δήλωσης του πεδίου και εισαγωγής των χωρικών δεδομένων σε αυτό. Για τη δεικτοδότηση των δεδομένων, χρησιμοποιούνται ευρετήρια τύπου GiST για το σύνολο των χωρικών οντοτήτων. Τέλος, το PostGIS παρέχει δυνατότητες προσπέλασης γεωμετρικών στοιχείων από Java Clients μέσω της διεπαφής JDBC (Java DataBase Connectivity) και οπτικοποίησης δεδομένων μέσω MapServer και uDIG [26]. Συνεπώς, τα δεδομένα μιας βάσης, η οποία είναι υλοποιημένη σύμφωνα με το πρότυπο (template) του PostGIS, είναι προσβάσιμα από χαρτογραφικές εφαρμογές, μέσα από την υλοποίηση ανάλογων προγραμματιστικών εφαρμογών.

Τυπικοί Χωρικοί Τελεστές	**Γεωμετρικές Συναρτήσεις**	**Συναρτήσεις Γεωμετρικού Ελέγχου**
DISJOINT	DISTANCE	SRID
CROSSES	AREA	Envelope
INTERSECTS	CENTROID	IsRing
EQUALS	LENGTH	IsClosed
TOUCHES	BOUNDARY	NumPoints
CONTAINS	BUFFER	EndPoint
WITHIN		
OVERLAPS		
RELATE		

Πίνακας 4-1: Βασικοί Τελεστές, Γεωμετρικές Συναρτήσεις και Συναρτήσεις Γεωμετρικού Ελέγχου του PostGIS

Πηγή: [26]

4.1.2 Το pgRouting

Το pgRouting συνιστά ένα εργαλείο επέκτασης των χωρικών λειτουργιών της PostgreSQL και του PostGIS, εμπλουτίζοντας τις δυνατότητές τους με την εισαγωγή συναρτήσεων οι οποίες επιτρέπουν τη διεξαγωγή χωρικών λειτουργιών δρομολόγησης σε δίκτυα. Το pgRouting παρέχει τη δυνατότητα οπτικοποίησης των χωρικών δεδομένων και των αποτελεσμάτων που προκύπτουν από την επεξεργασία τους, είτε μέσω των διεπαφών JDBC και ODBC είτε μέσω ερωτημάτων PL/pgSQL, σε διάφορα περιβάλλοντα χαρτογραφικής απεικόνισης, όπως το QGIS ή το uDIG. Παράλληλα, το υπολογιζόμενο κάθε φορά κόστος μιας διαδρομής υπολογίζεται δυναμικά και μπορεί να προκύπτει από το συνδυασμό πολλαπλών πεδίων ενός πίνακα ή ακόμη και από το συνδυασμό πεδίων που ανήκουν σε διαφορετικούς πίνακες της βάσης [URL9].

Οι συναρτήσεις που διαθέτει το pgRouting υλοποιούν μια σειρά αλγόριθμων δρομολόγησης που ο χρήστης μπορεί να επιλέγει κάθε φορά, για την εύρεση της ζητούμενης διαδρομής. Οι αλγόριθμοι που υλοποιούνται μέσω των συναρτήσεων του pgRouting είναι οι ακόλουθοι [URL9]:

- Ο αλγόριθμος εύρεσης συντομότερου μονοπατιού του Dijkstra, ο οποίος δε χρησιμοποιεί ευρετικούς κανόνες (heuristics).
- Ο αλγόριθμος εύρεσης συντομότερου μονοπατιού A*, ο οποίος χρησιμοποιείται σε περιπτώσεις δρομολόγησης σε μεγάλα δίκτυα και αξιοποιεί τα πλεονεκτήματα εφαρμογής ευρετικών κανόνων (heuristics).
- Ο αλγόριθμος εύρεσης συντομότερου μονοπατιού Shooting Star, ο οποίος χρησιμοποιείται σε περιπτώσεις δρομολόγησης από ακμή σε ακμή και όχι από κόμβο σε κόμβο όπως οι δύο προαναφερθέντες αλγόριθμοι και αξιοποιεί τα πλεονεκτήματα εφαρμογής ευρετικών κανόνων (heuristics).
- Ο αλγόριθμος Traveling Salesman Problem (tsp algorithm), ο οποίος εφαρμόζεται για την επίλυση του προβλήματος του πλανόδιου πωλητή.
- Ο αλγόριθμος Driving Distance Calculations (Isolines), ο οποίος χρησιμοποιείται για την ανεύρεση διαδρομών σε μικρής έκτασης τμήματα ενός δικτύου, όπως για παράδειγμα ένα τμήμα δικτύου που εκτείνεται 50 km γύρω από έναν κόμβο. Η συνάρτηση που υλοποιεί τον αλγόριθμο έχει ένα όρισμα που αφορά την απόσταση. Εάν κατά τη διαδικασία της αναζήτησης, η απόσταση/ κόστος που

έχει δηλωθεί αρχικά ξεπεραστεί τότε η εκτέλεση του αντίστοιχου ερωτήματος διακόπτεται.

Το pgRouting επεκτείνει τις δυνατότητες της PostgreSQL και του PostGIS, μέσα από την παροχή συναρτήσεων που επιτρέπουν την υλοποίηση χωρικών λειτουργιών σε δίκτυα. Ως εκ τούτου, ο χρήστης έχει τη δυνατότητα εύρεσης συντομότερων διαδρομών με εφαρμογή των αλγόριθμων δρομολόγησης που έχουν μοντελοποιηθεί με τη βοήθεια συναρτήσεων στο pgRouting και οπτικοποίησης των αποτελεσμάτων που προκύπτουν από το τρέξιμο του εκάστοτε αλγόριθμου, μέσα από τη σύνδεση του ΣΔΧΒΔ επάνω στο οποίο τρέχουν οι λειτουργίες του pgRouting με περιβάλλοντα χαρτογραφικής απεικόνισης.

4.2 Αναπαράσταση Οδικού Δικτύου στην PostgreSQL

Η εισαγωγή των στοιχείων ενός οδικού δικτύου σε μια χωρική βάση δεδομένων απαιτεί τη μοντελοποίησή τους, προκειμένου να καταστεί δυνατή η αναπαράστασή τους στη βάση με εφαρμογή χωρικών τύπων τους οποίους αναγνωρίζει το ΣΔΧΒΔ. Όπως έχει ήδη αναφερθεί σε προηγούμενες παραγράφους, ένα δίκτυο αναπαρίσταται με τη βοήθεια ενός γράφου και των στοιχείων που συνθέτουν το γράφο, δηλαδή των κόμβων και των ακμών που ορίζονται από τις σχέσεις συνδεσιμότητας μεταξύ των κόμβων του γράφου.

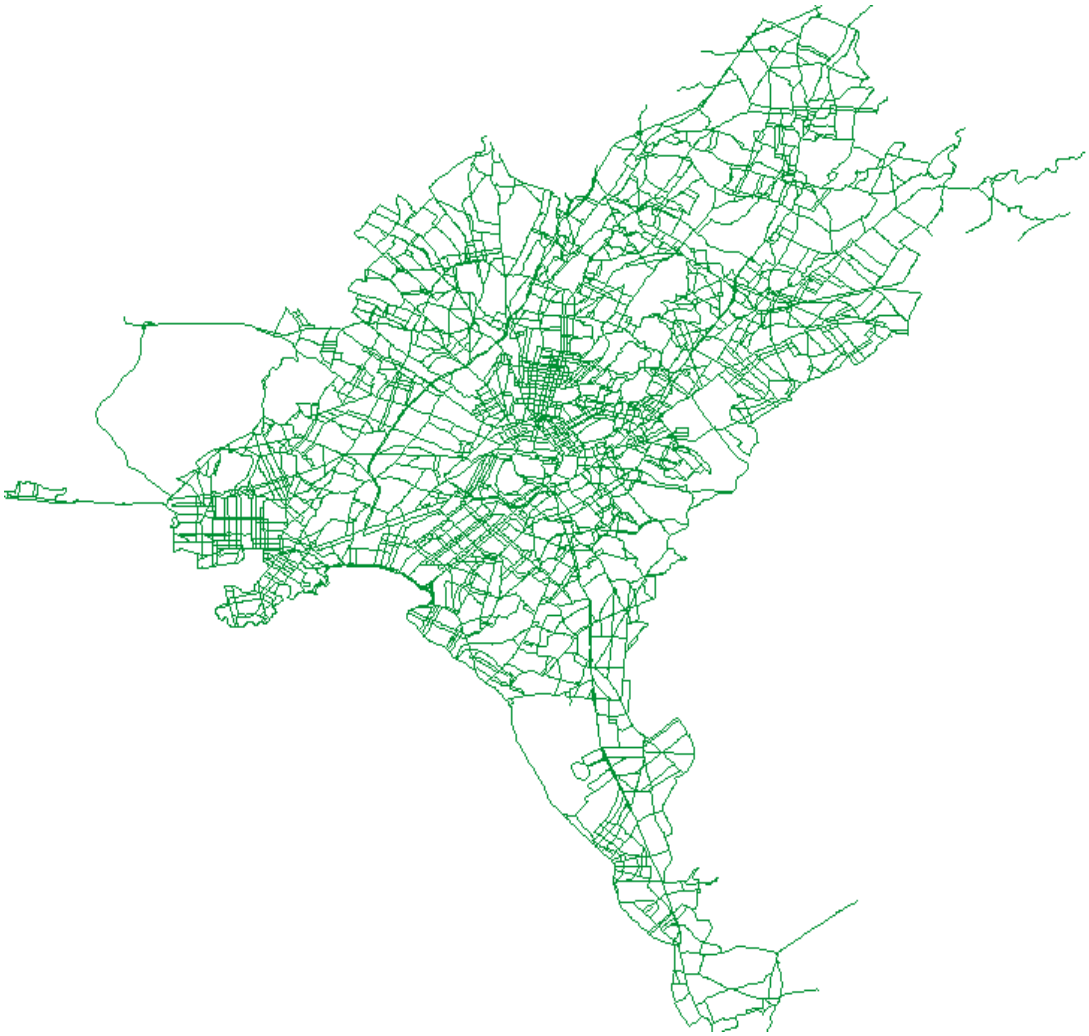
Στην περίπτωση μοντελοποίησης οδικών δικτύων, οι κόμβοι παριστάνουν συνήθως τις διασταυρώσεις του δικτύου και οι ακμές τμήματα του δικτύου, τα οποία υφίστανται μεταξύ των κόμβων. Τα στοιχεία του γράφου, αναπαρίστανται στη συνέχεια σε ένα ΣΔΧΒΔ ως μια ακολουθία συντεταγμένων που καταχωρείται στο πεδίο της γεωμετρίας του πίνακα που φιλοξενεί τα περιγραφικά και γεωγραφικά χαρακτηριστικά των χωρικών δεδομένων. Παράλληλα, ανάλογα με το ΣΔΧΒΔ που κάθε φορά χρησιμοποιείται για τη διαχείριση των χωρικών δεδομένων, καταχωρούνται και πρόσθετα στοιχεία που ορίζουν σαφέστερα τις χωρικές οντότητες όπως για παράδειγμα το σύστημα αναφοράς των δεδομένων, ο αριθμός των διαστάσεων των χωρικών οντοτήτων, ο τύπος των χωρικών οντοτήτων κ.ά.

Στη συνέχεια του παρόντος κεφαλαίου, παρουσιάζονται αναλυτικά οι διαδικασίες μοντελοποίησης και αναπαράστασης του βασικού οδικού δικτύου της Αθήνας στο περιβάλλον της PostgreSQL, καθώς και οι διαδικασίες δημιουργίας νέων συναρτήσεων

στη βάση, οι οποίες ανταποκρίνονται πλήρως στις ιδιαίτερες ανάγκες της παρούσας εφαρμογής.

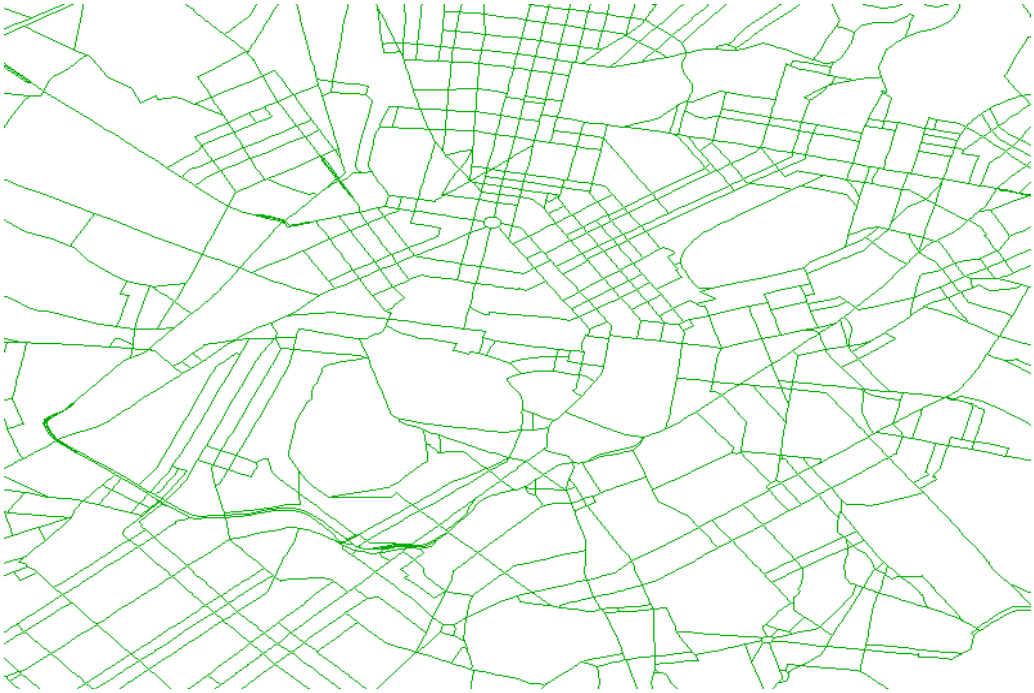
4.2.1 Δημιουργία χωρικής βάσης δεδομένων

Στα πλαίσια της παρούσας εφαρμογής, το οδικό δίκτυο της Αθήνας μοντελοποιήθηκε ως γράφος, οι κόμβοι του οποίου συνδέουν τους γεωμετρικούς συνδέσμους του δικτύου. Οι κόμβοι ορίζονται από τους κωδικούς και τις συντεταγμένες τους, ενώ τα τμήματα του οδικού δικτύου αναπαρίστανται ως ακολουθίες συντεταγμένων. Το σύστημα αναφοράς των χωρικών δεδομένων είναι το WGS84.



Σχήμα 4-1: Το Οδικό Δίκτυο της Αθήνας

Η δημιουργία της βάσης έγινε προγραμματιστικά με χρήση εντολών PL/SQL. Αρχικά, δημιουργήθηκε ο πίνακας που περιλαμβάνει τα περιγραφικά και γεωμετρικά χαρακτηριστικά των γεωγραφικών οντοτήτων του δικτύου και στη συνέχεια εισήχθησαν στον πίνακα τα περιγραφικά και γεωμετρικά στοιχεία του δικτύου.



Σχήμα 4-2: Τμήμα του Οδικού Δικτύου της Αθήνας

Η PL/SQL, επεκτείνει τις δυνατότητες της SQL παρέχοντας δυνατότητες χρήσης μεταβλητών, χειρισμού σφαλμάτων και εξαιρέσεων, εισαγωγής επαναληπτικών βρόχων σε προγράμματα κ.λπ. Έτσι, καθίσταται δυνατή η δημιουργία αμιγώς προγραμματιστικών τμημάτων κώδικα για την εκτέλεση πάσης φύσεως λειτουργιών στη βάση. Η βασική δομή ενός κώδικα PL/SQL περιλαμβάνει τέσσερα βασικά τμήματα κώδικα, το δηλωτικό το οποίο ορίζεται με την εντολή *DECLARE* και περιλαμβάνει τον ορισμό μεταβλητών και τύπων μεταβλητών, το τμήμα εκτέλεσης του κώδικα το οποίο ορίζεται με την εντολή *BEGIN* και περιλαμβάνει τις επιμέρους διαδικασίες και εντολές του κώδικα PL/SQL, το τμήμα χειρισμού εξαιρέσεων το οποίο ορίζεται με την εντολή *EXCEPTION* και περιλαμβάνει εντολές χειρισμού σφαλμάτων και το τμήμα όπου δηλώνεται το τέλος του κώδικα με την εντολή *END*. Η γλώσσα PL/SQL δημιουργήθηκε αρχικά για την εκτέλεση λειτουργιών στην Oracle, αλλά στη συνέχεια η χρήση της επεκτάθηκε ούτως ώστε να μπορεί να χρησιμοποιηθεί και σε άλλα περιβάλλοντα ανάπτυξης βάσεων δεδομένων. Η

PostgreSQL χρησιμοποιεί μια ειδική έκδοση της PL/SQL, την PL/pgSQL, η οποία διαφοροποιείται μερικώς από την PL/SQL προκειμένου να προσαρμόζεται σε μεγαλύτερο βαθμό στο ιδιαίτερο περιβάλλον της PostgreSQL. Οι βασικές διαφοροποιήσεις που εμφανίζει η PL/pgSQL σε σχέση με την PL/SQL, εντοπίζονται στα παρακάτω βασικά σημεία [URL14]:

- Στην PostgreSQL, δεν υπάρχουν προκαθορισμένες τιμές (default values) για τις επιμέρους παραμέτρους των κωδίκων PL/pgSQL.
- Στην PL/pgSQL αντί για τη χρήση κέρσορα, ο χρήστης μπορεί να εισάγει το ερώτημά του μέσα σε έναν επαναληπτικό βρόχο *FOR*.
- Η PostgreSQL, εμφανίζει κάποιες ιδιαιτερότητες ως προς τον αριθμό των quotes που απαιτούνται κατά τη διαχείριση αλφαριθμητικών τιμών (strings) μέσα στο κύριο τμήμα ορισμού μιας συνάρτησης. Ανάλογα με το κάθε φορά επιδιωκώμενο αποτέλεσμα υπάρχουν αντίστοιχοι κανόνες σύνταξης.

Στο πρώτο στάδιο υλοποίησης της παρούσας εφαρμογής, δημιουργήθηκε μια νέα χωρική βάση δεδομένων, η ‘testgis’ σύμφωνα με το πρότυπο του PostGIS (template PostGIS), προκειμένου να εμπλουτιστεί με τις πρόσθετες λειτουργίες διαχείρισης χωρικών δεδομένων που διαθέτει το PostGIS. Στη συνέχεια, ακολούθησε η δημιουργία του πίνακα της βάσης και η εισαγωγή των χωρικών δεδομένων με τη βοήθεια κώδικα PL/pgSQL. Ο πίνακας της βάσης δημιουργήθηκε με την εντολή *CREATE TABLE* και αποτελείται από 22 πεδία στα οποία έχει καταχωρηθεί η απαραίτητη πληροφορία που αφορά το χωρικό δίκτυο. Το ερώτημα βάσει του οποίου δημιουργήθηκε ο πίνακας είναι το ακόλουθο:

```
CREATE TABLE "athens" (gid serial PRIMARY KEY, "length" float8, "id" int8,
"greekname" varchar(50), "latinname" varchar(50), "class_id" int2, "ring" int2,
"oneway" varchar(2), "ft_speed" int2, "tf_speed" int2, "source" int4, "x1" float8,
"y1" float8, "target" int4, "x2" float8, "y2" float8, "flag" int2, "cost" float8,
"reverse_co" float8);
```

Οι παραπάνω εντολές συνιστούν τμήμα του PL/pgSQL κώδικα που έτρεξε για τη δημιουργία του πίνακα ‘athens’ και την εισαγωγή των χωρικών δεδομένων. Αναλυτικότερα, σε κάθε πεδίο του πίνακα καταχωρείται η ακόλουθη περιγραφική πληροφορία:

- **gid**: Κωδικός ταυτοποίησης κάθε γεωμετρικής οντότητας και πρωτεύον κλειδί των εγγραφών.
- **length**: Γεωμετρικό μήκος κάθε τμήματος του οδικού δικτύου που καταχωρείται στη στήλη της γεωμετρίας, σε μέτρα.
- **id**: Αύξων αριθμός γεωμετρικού συνδέσμου.
- **greekname**: Ονομασία δρόμου που περιλαμβάνει τον εκάστοτε γεωμετρικό σύνδεσμο, στα ελληνικά.
- **latinname**: Ονομασία δρόμου που περιλαμβάνει τον εκάστοτε γεωμετρικό σύνδεσμο, στα αγγλικά.
- **class_id**: Κωδικός κατηγορίας οδικού δικτύου (1: αυτοκινητόδρομος, 2: οδικός άξονας ταχείας κυκλοφορίας, 3: κύρια αρτηρία, 4: δευτερεύουσα αρτηρία, 5: συλλεκτήρια οδός).
- **ring**: Ένδειξη δακτυλίου (0: εκτός δακτυλίου, 1: εντός δακτυλίου, 2: όριο δακτυλίου).
- **oneway**: Χαρακτηρισμός μονοδρόμων (N: ο γεωμετρικός σύνδεσμος δεν είναι μονόδρομος, Y: ο γεωμετρικός σύνδεσμος είναι μονόδρομος).
- **ft_speed**: Μέση ταχύτητα κίνησης των οχημάτων κατά τη φορά της ψηφιοποίησης σε km/h.
- **tf_speed**: Μέση ταχύτητα κίνησης των οχημάτων σε φορά αντίθετη από τη φορά της ψηφιοποίησης σε km/h.
- **source**: Κωδικός κόμβου αφετηρίας γεωμετρικού συνδέσμου.
- **x1**: Συντεταγμένη x του κόμβου αφετηρίας γεωμετρικού συνδέσμου.
- **y1**: Συντεταγμένη y του κόμβου αφετηρίας γεωμετρικού συνδέσμου.
- **target**: Κωδικός κόμβου προορισμού γεωμετρικού συνδέσμου.
- **x2**: Συντεταγμένη x του κόμβου προορισμού γεωμετρικού συνδέσμου.
- **y2**: Συντεταγμένη y του κόμβου προορισμού γεωμετρικού συνδέσμου.
- **flag**: Ένδειξη σήμανσης γεωμετρικού συνδέσμου.
- **cost**: Απαιτούμενο κόστος για τη διάσχιση ενός γεωμετρικού συνδέσμου κατά τη φορά της ψηφιοποίησης, σε μονάδες χρόνου (δευτερόλεπτα).

- **reverse_cost**: Απαιτούμενο κόστος για τη διάσχιση ενός γεωμετρικού συνδέσμου με φορά αντίθετη από αυτή της ψηφιοποίησης, σε μονάδες χρόνου (δευτερόλεπτα).

Στη συνέχεια, ακολουθεί η δημιουργία του πεδίου της γεωμετρίας στο οποίο πρόκειται να καταχωρηθούν οι γεωμετρικοί σύνδεσμοι του οδικού δικτύου που έχουν μοντελοποιηθεί ως ακμές ενός γράφου και αναπαρίστανται ως μια ακολουθία συντεταγμένων. Η δημιουργία του πεδίου της γεωμετρίας στο PostGIS, ορίζεται με τη βοήθεια της συνάρτησης *AddGeometryColumn()*, η οποία δέχεται ως ορίσματα το όνομα του σχήματος της βάσης, το όνομα του πίνακα στον οποίο πρόκειται να δημιουργηθεί το νέο γεωμετρικό πεδίο, το όνομα του γεωμετρικού πεδίου, ο κωδικός του συστήματος αναφοράς των δεδομένων, ο τύπος των χωρικών δεδομένων που πρόκειται να καταχωρηθούν στο γεωμετρικό πεδίο και τέλος το πλήθος των διαστάσεων των γεωμετρικών οντοτήτων. Η συνάρτηση *AddGeometryColumn()*, συντάσσεται ως ακολούθως:

```
SELECT
AddGeometryColumn('','athens','the_geom','4326','MULTILINESTRING',2);
```

Στο σημείο αυτό, θα πρέπει να σημειωθεί ότι όταν το αναφερόμενο σχήμα της βάσης είναι το *'public'*, το αντίστοιχο όρισμα της συνάρτησης μπορεί να παραμείνει κενό, καθώς εάν δεν προσδιορίζεται κάποιο άλλο σχήμα, το PostGIS δημιουργεί αυτόματα το νέο πεδίο στον αναφερόμενο πίνακα του σχήματος *'public'*.

Κατά τη διαδικασία δημιουργίας πινάκων σε μια χωρική βάση δεδομένων, προηγούνται οι εντολές δημιουργίας των πεδίων που φιλοξενούν τα περιγραφικά χαρακτηριστικά των χωρικών δεδομένων και ακολουθούν οι εντολές δημιουργίας του γεωμετρικού πεδίου στο οποίο καταχωρούνται τα γεωμετρικά χαρακτηριστικά των γεωγραφικών οντοτήτων. Στην περίπτωση της παρούσας εφαρμογής, το πεδίο της γεωμετρίας ονομάζεται *'the_geom'*, το σύστημα αναφοράς είναι το WGS84 ο κωδικός του οποίου στον πίνακα συστημάτων αναφοράς του PostGIS είναι ο *4326*, ο τύπος που αναπαριστά τα χωρικά δεδομένα είναι η πολυγραμμή *(multilinestring)* και τέλος, ο αριθμός *2* υποδεικνύει ότι τα χωρικά δεδομένα είναι δυδιάστατα.

Ακολούθησε η εισαγωγή των δεδομένων στη βάση με τη βοήθεια των εντολών *INSERT*. Η διαδικασία δημιουργίας του πίνακα και εισαγωγής των χωρικών δεδομένων σε αυτόν, ολοκληρώθηκε όταν εκτελέστηκε το σύνολο των εντολών του τμήματος *BEGIN* του PL/pgSQL κώδικα.

Στη συνέχεια, ορίστηκαν τα ευρετήρια για την οργάνωση των χωρικών δεδομένων και την ταχύτερη προσπέλασή τους και υλοποιήθηκαν ορισμένοι έλεγχοι που εκτελεί το PostGIS με τη βοήθεια συναρτήσεων γεωμετρικού ελέγχου προκειμένου να αξιολογηθεί η ορθότητα της γεωμετρίας των δεδομένων. Τα ευρετήρια ορίστηκαν στις στήλες *source* και *target* του πίνακα και η δομή που χρησιμοποιήθηκε για τη δεικτοδότηση των δεδομένων είναι το B- δένδρο (btree). Τέλος, οι γεωμετρικοί έλεγχοι που υλοποιήθηκαν, αφορούν τον έλεγχο της διάστασης των χωρικών δεδομένων, τον έλεγχο του τύπου που χρησιμοποιήθηκε για την αναπαράσταση των χωρικών δεδομένων και τον έλεγχο του συστήματος αναφοράς.

4.2.2 Αλγόριθμοι και συναρτήσεις δρομολόγησης

Στο στάδιο αυτό, ακολούθησε το τρέξιμο των συναρτήσεων δρομολόγησης που διαθέτει το pgRouting προκειμένου να ελεγχθεί η συμπεριφορά του δικτύου και να αξιολογηθούν τα αποτελέσματα που προκύπτουν από τους υπολογισμούς συντομότερης διαδρομής. Από το σύνολο των συναρτήσεων που μοντελοποιούν αλγόριθμους δρομολόγησης, δοκιμάστηκαν αλγόριθμοι που αφορούν την αναζήτηση συντομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου.

Πριν το τρέξιμο των αλγόριθμων συντομότερης διαδρομής, δημιουργήθηκε ένα *SEQUENCE* προκειμένου τα αποτελέσματα που προκύπτουν από τους υπολογισμούς συντομότερων διαδρομών να βρίσκονται ταξινομημένα. Με τον τρόπο αυτό, τα οδικά τμήματα που συνθέτουν κάθε προτεινόμενη διαδρομή, παρουσιάζονται κατά τη σειρά με την οποία πρέπει να διασχιστούν προκειμένου ο χρήστης της εφαρμογής να μεταβεί από το σημείο εκκίνησης στο σημείο προορισμού και όχι αταξινόμητα. Ουσιαστικά, η δημιουργία ενός *SEQUENCE* έχει ως αποτέλεσμα τη δημιουργία ενός «μετρητή» που ταξινομεί δεδομένα σε μια σειρά, ταυτοποιώντας τις εγγραφές του εκάστοτε πίνακα που περιέχει τα δεδομένα. Πρόκειται για τη δημιουργία ενός ειδικού μονοδιάστατου πίνακα, επάνω στον οποίο τρέχουν ειδικές συναρτήσεις μέσω των οποίων ο πίνακας αυτός καθίσταται διαχειρίσιμος. Στην προκειμένη περίπτωση, η συνάρτηση που έτρεξε επάνω στο *SEQUENCE* που δημιουργήθηκε ήταν η συνάρτηση *nextval()* η οποία «μετρά» τα αντικείμενα που εισέρχονται στο *SEQUENCE* και τα ταξινομεί.

Αρχικά, δοκιμάστηκαν οι έτοιμες συναρτήσεις που διαθέτει το pgRouting και οι οποίες υλοποιούν τον αλγόριθμο του Dijkstra, τον αλγόριθμο A* και τον αλγόριθμο Traveling Sales Person. Τα αποτελέσματα των συναρτήσεων αφορούν την εκάστοτε υπολογισμένη συντομότερη διαδρομή, η οποία περιγράφεται από τους κόμβους που ορίζουν τα γεωμετρικά τμήματα που συνθέτουν τη διαδρομή, τους κωδικούς κάθε γεωμετρικού συνδέσμου και το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου σε μονάδες χρόνου. Ωστόσο, οι ιδιαίτερες απαιτήσεις της εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας οδήγησαν στη δημιουργία νέων συναρτήσεων, οι οποίες συνιστούν κατά κάποιο τρόπο παραλλαγή των ήδη υφιστάμενων συναρτήσεων του pgRouting. Οι νέες συναρτήσεις ανταποκρίνονται πλήρως στις απαιτήσεις της εφαρμογής.

Οι λόγοι που επέβαλαν τη δημιουργία νέων συναρτήσεων, συνίστανται κατά βάση στον τύπο των αποτελεσμάτων που πρέπει να προκύπτουν μετά το τρέξιμο κάθε αλγόριθμου δρομολόγησης. Στα αποτελέσματα κάθε συνάρτησης, πρέπει να περιλαμβάνεται ο κωδικός κάθε γεωμετρικού τμήματος που ανήκει στο συντομότερο μονοπάτι, η ονομασία κάθε γεωμετρικού τμήματος με ελληνικούς και λατινικούς χαρακτήρες, το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου καθώς και η γεωμετρία κάθε συνδέσμου, στοιχεία τα οποία δεν εξάγονται συνολικά από τις ήδη υφιστάμενες υλοποιημένες συναρτήσεις του pgRouting. Επιπλέον, το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου θα πρέπει να υπολογίζεται είτε βάσει του χρόνου είτε βάσει της γεωμετρικής απόστασης του συνδέσμου, δυνατότητα η οποία δεν παρέχεται από τις υπάρχουσες συναρτήσεις καθώς αυτές υπολογίζουν το κόστος βάσει της στήλης *'cost'* του πίνακα της βάσης στην οποία, περιλαμβάνεται μεν το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου αλλά αποκλειστικά και μόνο σε μονάδες χρόνου. Όσον αφορά τη γεωμετρία κάθε γεωμετρικού συνδέσμου θα πρέπει να εξάγεται υπό μορφή *kml*, ούτως ώστε να είναι δυνατή η απεικόνισή της στο διαδικτυακό χάρτη της ιστοσελίδας, γεγονός που δεν καθίσταται δυνατό μέσα από τη χρήση των υφιστάμενων συναρτήσεων. Τέλος, τα οδικά τμήματα που συνθέτουν την εκάστοτε συντομότερη διαδρομή που υπολογίζεται, θα πρέπει να είναι ταξινομημένα σύμφωνα με τη σειρά που πρέπει να διασχιστούν από το σημείο αφετηρίας μέχρι το σημείο προορισμού.

Τα παραπάνω ζητήματα επιλύθηκαν με τις αντίστοιχες παρεμβάσεις στους κώδικες των υφιστάμενων συναρτήσεων, ούτως ώστε κάθε φορά να προκύπτει το επιθυμητό αποτέλεσμα σύμφωνα με τις ιδιαίτερες απαιτήσεις της εφαρμογής. Στις παραγράφους που ακολουθούν, παρουσιάζονται αναλυτικά οι παρεμβάσεις που έγιναν ανά περίπτωση, ενώ

οι κώδικες των νέων συναρτήσεων που προέκυψαν παρατίθενται στο Παράρτημα του παρόντος τεύχους.

Ο αλγόριθμος Dijkstra

Ο πρώτος αλγόριθμος που δοκιμάστηκε, είναι ο αλγόριθμος του Dijkstra. Το pgRouting διαθέτει μια σειρά από συναρτήσεις που μοντελοποιούν τον αλγόριθμο του Dijkstra και εφαρμόζονται ανάλογα με το πρόβλημα που τίθεται κάθε φορά προς επίλυση αλλά και τη μορφή του γράφου. Οι συναρτήσεις αυτές διαφοροποιούνται μεταξύ τους ως προς τα ορίσματα που δέχονται και τα αποτελέσματα που προκύπτουν μετά το πέρας των αντίστοιχων υπολογισμών. Σε κάθε περίπτωση, προκύπτει η συντομότερη διαδρομή για τη μετάβαση από ένα σημείο του δικτύου σε κάποιο άλλο. Η γενική συνάρτηση που διαθέτει το pgRouting για την εύρεση της συντομότερης διαδρομής μεταξύ ενός κόμβου αφετηρίας και ενός κόμβου προορισμού, με εφαρμογή του αλγόριθμου του Dijkstra, είναι η συνάρτηση *shortest_path()*. Το αποτέλεσμα της συνάρτησης αυτής περιλαμβάνει τους κωδικούς των κόμβων αφετηρίας των ακμών που συνθέτουν τη συντομότερη διαδρομή που κάθε φορά προκύπτει *(vertex_id)*, τους κωδικούς αναγνώρισης των ακμών που ανήκουν στο συγκεκριμένο μονοπάτι *(edge_id)* καθώς και το κόστος διάσχισης κάθε ακμής *(cost)* σε μονάδες χρόνου. Μία δεύτερη συνάρτηση που διαθέτει το pgRouting και αφορά τον αλγόριθμο του Dijkstra είναι η *dijkstra_sp_directed()*, η οποία δίνει ως αποτέλεσμα τον κωδικό και τον αύξοντα αριθμό κάθε γεωμετρικού συνδέσμου καθώς και τη γεωμετρία του κάθε γεωμετρικού συνδέσμου ως μια ακολουθία συντεταγμένων *(multilinestring)*. Η συνάρτηση αυτή τρέχει επάνω στη *shortest_path()* και χρησιμοποιείται για την υποστήριξη διαδικασιών οπτικοποίησης των συντομότερων μονοπατιών στο QGIS.

Το PostGIS επιτρέπει στο χρήστη τη σύνδεση της βάσης δεδομένων με λογισμικά οπτικοποίησης, όπως το QGIS. Στην περίπτωση που ο χρήστης επιθυμεί να δει οπτικοποιημένα τα συντομότερα μονοπάτια που προκύπτουν μετά το τρέξιμο κάθε αλγόριθμου, το PostGIS επιτρέπει τη δημιουργία νέων πινάκων που περιλαμβάνουν τα αποτελέσματα, τα οποία στη συνέχεια και αφού επιτευχθεί η σύνδεση της βάσης με το QGIS, αναπαρίστανται γραφικά στο QGIS.

Τέλος, οι συναρτήσεις *dijkstra_sp_delta()* και *dijkstra_sp_delta_directed()*, αφορούν και πάλι στην αναζήτηση συντομότερων μονοπατιών με εφαρμογή του αλγόριθμου του Dijkstra κατ' αντιστοιχία με τις δύο προηγούμενες συναρτήσεις, περιορίζοντας όμως την αναζήτηση σε ένα συγκεκριμένο τμήμα του δικτύου και όχι στο σύνολό του. Η

συναρτήσεις αυτές είναι ιδιαίτερα αποτελεσματικές σε περιπτώσεις μεγάλων δικτύων, καθώς μειώνεται ο χρόνος αναζήτησης και ο όγκος των δεδομένων που τίθενται προς επεξεργασία.

Παρά το γεγονός ότι οι συναρτήσεις του pgRouting δίνουν τα συντομότερα μονοπάτια μεταξύ δύο κόμβων, το είδος της εξαγόμενης πληροφορίας που απαιτείται στα πλαίσια της σχεδιαζόμενης εφαρμογής οδήγησε στη δημιουργία μιας νέας συνάρτησης. Η συνάρτηση αυτή επεκτείνει τις λειτουργίες μίας ήδη υλοποιημένης συνάρτησης του pgRouting, δίνοντας λεπτομερέστερη πληροφορία για τη συντομότερη διαδρομή που προκύπτει μετά το τρέξιμο του αλγόριθμου του Dijkstra. Πιο αναλυτικά, ο χρήστης της ιστοσελίδας που σχεδιάστηκε, οι λειτουργίες της οποίας απαιτούν για την ολοκλήρωσή τους την άντληση πληροφορίας από τη βάση δεδομένων, θα πρέπει να λαμβάνει το αποτέλεσμα για τη συντομότερη διαδρομή που αναζητά, τόσο ως λεκτική περιγραφή όσο και ως γεωμετρική πολυγραμμή που θα απεικονίζεται επάνω σε διαδικτυακό χάρτη. Η λεκτική περιγραφή της διαδρομής που προκύπτει μετά το τρέξιμο του αλγόριθμου, αφορά στις ονομασίες των οδών που περιλαμβάνονται στην εκάστοτε διαδρομή και στο κόστος διάσχισης κάθε οδικού τμήματος. Η γεωμετρική περιγραφή αφορά στη διανυσματική απεικόνιση της διαδρομής, η οποία θα προκύπτει ως αποτέλεσμα της μετατροπής της γεωμετρίας των χωρικών δεδομένων σε *kml format*.

Η συνάρτηση που δημιουργήθηκε ονομάστηκε *dijkstra_sp_directed_cost()*. Σχεδιάστηκε σύμφωνα με το πρότυπο της ήδη υπάρχουσας συνάρτησης *dijkstra_sp_directed()* και ο σχεδιασμός της υλοποιήθηκε σε γλώσσα PL/pgSQL. Ο κώδικας της συνάρτησης βρίσκεται στο Παράρτημα. Θα πρέπει να σημειωθεί ότι για τη δημιουργία της νέας συνάρτησης, απαιτήθηκε πρώτα η δημιουργία ενός νέου τύπου δεδομένων ο οποίος περιγράφει τον τύπο του αποτελέσματος που επιστρέφει η νέα συνάρτηση. Ο νέος τύπος που δημιουργήθηκε, είναι ο τύπος *geomsc* και ο κώδικας δημιουργίας του βρίσκεται στο Παράρτημα. Η νέα συνάρτηση δίνει ως αποτέλεσμα τον αύξοντα αριθμό *(id)* κάθε γεωμετρικού συνδέσμου που ανήκει στο συντομότερο μονοπάτι που κάθε φορά προκύπτει από το τρέξιμο του αλγόριθμου, τον κωδικό ταυτοποίησης των γεωμετρικών συνδέσμων *(gid*), το όνομα κάθε γεωμετρικού συνδέσμου στα ελληνικά *(greekname)*, το όνομα κάθε γεωμετρικού συνδέσμου στα λατινικά *(latinname)*, το κόστος *(cost)* διάσχισης κάθε οδικού συνδέσμου και τη γεωμετρία κάθε γεωμετρικού συνδέσμου σε μορφή *kml*. Όσον αφορά το κόστος, υπάρχει δυνατότητα υπολογισμού του είτε σύμφωνα με τη γεωμετρική απόσταση του εκάστοτε γεωμετρικού συνδέσμου, είτε σύμφωνα με το χρόνο που απαιτείται για τη διάσχισή του. Συνεπώς, όπως αναλύεται σαφέστερα σε επόμενο κεφάλαιο, ο χρήστης της εφαρμογής

επιλέγει τον τρόπο υπολογισμού του κόστους και η βάση ανταποκρίνεται ανάλογα. Όσον αφορά τη γεωμετρία, μετατρέπεται σε *kml format* με χρήση της συνάρτησης *ST_AsKML()*, η οποία τρέχει επάνω στη συνάρτηση *dijkstra_sp_directed_cost()* και μετατρέπει τα δεδομένα του πεδίου της γεωμετρίας *'the_geom'* του πίνακα *'athens'* σε μορφή *kml*. Τα αποτελέσματα που προκύπτουν, ταξινομούνται βάσει του *SEQUENCE* που έχει υλοποιηθεί και συμμετέχει στη διαδικασία ως όρισμα της συνάρτησης με την ονομασία *'resNodes'*. Στη συνέχεια, δίνονται δύο ενδεικτικά παραδείγματα όπου παρουσιάζεται η λειτουργία της συνάρτησης *dijkstra_sp_directed_cost()* και κάποια ενδεικτικά αποτελέσματα.

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM
dijkstra_sp_directed_cost('athens', 3186, 5154, true, true, 'length', 'resNodes');
```

Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	8842	305.757	ΔΕΚΕΛΕΙΑΣ ΛΕΩΦ.	DEKELEIAS	<MultiGeometry><LineString><coordinates>23.7468471759461,38.0482198(
2	2	8789	58.9361	ΔΕΚΕΛΕΙΑΣ ΛΕΩΦ.	DEKELEIAS	<MultiGeometry><LineString><coordinates>23.74914938279 72,38.0502842
3	3	8788	26.7338	ΙΦΙΓΕΝΕΙΑΣ	IFIGENEIAS	<MultiGeometry><LineString><coordinates>23.7512754163825,38.0503609(
4	4	8731	83.5373	ΠΕΥΚΩΝ	PEFKON	<MultiGeometry><LineString><coordinates>23.7563633065331,38.0555630(
5	5	8625	18.951	ΠΕΥΚΩΝ	PEFKON	<MultiGeometry><LineString><coordinates>23.7570233419003,38.0566278
6	6	8593	21.2967	ΠΕΥΚΩΝ	PEFKON	<MultiGeometry><LineString><coordinates>23.7578953881189,38.0578914

Πίνακας 4-2: Ενδεικτικά Αποτελέσματα Συνάρτησης dijkstra_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Γεωμετρικής Απόστασης

Στο παραπάνω παράδειγμα, διατυπώνεται ένα ερώτημα εύρεσης συντομότερης διαδρομής με εφαρμογή της συνάρτησης *dijkstra_sp_directed_cost()*. Στον πίνακα 4-2, φαίνονται οι έξι πρώτοι γεωμετρικοί σύνδεσμοι του συντομότερου μονοπατιού που προέκυψε από το τρέξιμο του αλγόριθμου του Dijkstra. Τα ορίσματα που δέχεται η συνάρτηση είναι το όνομα του πίνακα που βρίσκονται αποθηκευμένα τα στοιχεία του δικτύου, το σημείο (κόμβος) εκκίνησης της διαδρομής, το σημείο (κόμβος) προορισμού της διαδρομής, μια λογική τιμή (boolean value) που ορίζει εάν το δίκτυο είναι κατευθυνόμενο *(dir= true)* ή μη κατευθυνόμενο *(dir=false)*, μια λογική τιμή που καθορίζει εάν στους υπολογισμούς θα ληφθεί υπόψη το κόστος διάσχισης μιας διαδρομής

προς την αντίθετη κατεύθυνση *(reverse_cost)*, το κόστος διάσχισης της διαδρομής, το οποίο στην προκειμένη περίπτωση υπολογίζεται σύμφωνα με τη γεωμετρική απόσταση των γεωμετρικών συνδέσμων και υποδεικνύεται με το *'length'* και τέλος το όνομα του *SEQUENCE* σύμφωνα με το οποίο θα ταξινομηθούν με την ορθή σειρά διάσχισης οι γεωμετρικοί σύνδεσμοι.

Στο επόμενο παράδειγμα, φαίνονται τα αποτελέσματα από το τρέξιμο της ίδιας συνάρτησης, με τη διαφορά ότι το όρισμα που αφορά το κόστος ορίζεται στην περίπτωση αυτή ως *'cost'*, το οποίο υποδεικνύει τον υπολογισμό του κόστους διάσχισης της διαδρομής σε μονάδες χρόνου.

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM
dijkstra_sp_directed_cost('athens',3186,5154, true, true, 'cost', 'resNodes');
```

Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	8842	68.7953	ΔΕΚΕΛΕΙΑΣ ΛΕΩΦ.	DEKELEIAS	<MultiGeometry><LineString><coordinates>23.7468471759461,38.0482198
2	2	8789	17.6808	ΔΕΚΕΛΕΙΑΣ ΛΕΩΦ.	DEKELEIAS	<MultiGeometry><LineString><coordinates>23.7491493827972,38.0502842
3	3	8667	108.253	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.7495678744434,38.0506997
4	4	8542	21.2051	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.7549500569301,38.0600934
5	5	8512	54.2945	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.7549500569301,38.0600934
6	6	8464	36.9006	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.757118155651,38.0624954

Πίνακας 4-3: Ενδεικτικά Αποτελέσματα Συνάρτησης dijkstra_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Χρόνου Διάσχισης

*Ο αλγόριθμος A**

Ο επόμενος αλγόριθμος που έτρεξε για την εύρεση συντομότερης διαδρομής στο δίκτυο είναι ο αλγόριθμος astar (A*). Ο αλγόριθμος A* χρησιμοποιεί ευρετικούς κανόνες (heuristics) για την εύρεση συντομότερης διαδρομής σε μεγάλα δίκτυα, προκειμένου να μειωθεί ο απαιτούμενος χρόνος αναζήτησης. Η ολοκλήρωση μιας διαδικασίας εύρεσης συντομότερης διαδρομής με εφαρμογή του αλγόριθμου A* απαιτεί συνήθως λιγότερο

χρόνο σε σχέση με τον αλγόριθμο του Dijkstra. Στην περίπτωση του δικτύου της Αθήνας, οι χρονικές διαφορές μεταξύ των δύο αλγόριθμων είναι αμελητέες, καθώς δεν πρόκειται για ένα πολύ μεγάλης έκτασης δίκτυο. Ενδεικτικά, αναφέρεται ότι για το τρέξιμο του ίδιου τύπου ερωτήματος και για την εύρεση συντομότερης διαδρομής μεταξύ των ίδιων κόμβων αφετηρίας και προορισμού, ο αλγόριθμος του Dijkstra έτρεξε σε χρόνο 79 ms ενώ ο A* σε χρόνο ίσο με 78 ms.

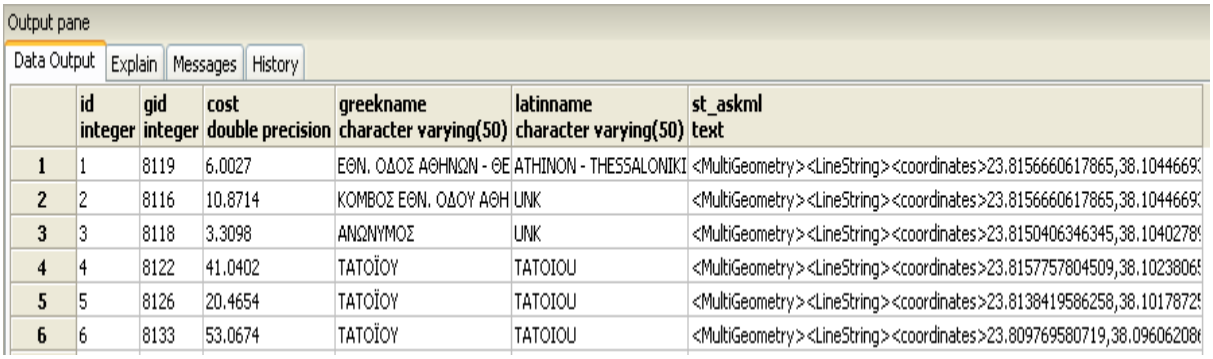
Όμοια με τον αλγόριθμο του Dijkstra, το pgRouting διαθέτει συναρτήσεις που υλοποιούν τον αλγόριθμο A* για την αναζήτηση συντομότερης διαδρομής. Η βασική συνάρτηση που αφορά τον αλγόριθμο A* είναι η *shortest_path_astar()*, αντίστοιχη με τη συνάρτηση *shortest_path()* που υλοποιεί τον αλγόριθμο του Dijkstra. Η συνάρτηση *shortest_path_astar()* δίνει ως αποτέλεσμα τους κωδικούς των κόμβων αφετηρίας των ακμών που συνθέτουν τη συντομότερη διαδρομή που κάθε φορά προκύπτει *(vertex_id)*, τους κωδικούς αναγνώρισης των ακμών που ανήκουν στο συγκεκριμένο μονοπάτι *(edge_id)* καθώς και το κόστος διάσχισης κάθε ακμής *(cost)*. Η συνάρτηση *astar_sp_directed()* είναι μια δεύτερη συνάρτηση που υλοποιεί τον αλγόριθμο A* και παρέχει, όπως και η αντίστοιχη συνάρτηση για το Dijkstra, τη δυνατότητα οπτικοποίησης των αποτελεσμάτων που προκύπτουν από το τρέξιμο του αλγόριθμου A* σε περιβάλλοντα χαρτογραφικής απεικόνισης, όπως το QGIS. Παράλληλα, διατίθεται μια σειρά συναρτήσεων που υλοποιούν τον αλγόριθμο A*, περιορίζοντας πρώτα την έκταση της περιοχής αναζήτησης με δημιουργία *bounding boxes*, όπως οι συναρτήσεις *astar_sp_bbox_directed()* και *astar_sp_delta_directed()*. Οι συναρτήσεις αυτές δημιουργούν, με διαφορετικό τρόπο η κάθε μία, *bounding boxes* και περιορίζουν την έκταση της αναζήτησης. Η συνάρτηση *astar_sp_delta_directed()* για παράδειγμα δημιουργεί ένα *bounding box*, μέσα από τον προσδιορισμό της απόστασης του κόμβου αφετηρίας και του κόμβου προορισμού από το πλαίσιο του ορθογωνίου που περικλείει την περιοχή ενδιαφέροντος.

Οι απαιτήσεις της εφαρμογής οδήγησαν και στην περίπτωση του αλγόριθμου A* στη δημιουργία μιας νέας συνάρτησης, σύμφωνα με το πρότυπο ήδη υπάρχουσας συνάρτησης που υλοποιεί τον αλγόριθμο, προκειμένου το εξαγόμενο αποτέλεσμα να είναι το επιθυμητό από το χρήστη της εφαρμογής. Η συνάρτηση που δημιουργήθηκε, παρέχει τον ίδιο τύπο πληροφορίας με την αντίστοιχη συνάρτηση που υλοποιήθηκε για τον αλγόριθμο του Dijkstra. Είναι η συνάρτηση *astar_sp_directed_cost()* και επεκτείνει τις δυνατότητες της ήδη υπάρχουσας συνάρτησης *astar_sp_directed()*. Η συνάρτηση *astar_sp_directed()* δέχεται ως ορίσματα το όνομα του πίνακα της βάσης που περιλαμβάνει τα στοιχεία του δικτύου, τους κωδικούς των κόμβων αφετηρίας και προορισμού και δύο λογικές τιμές

(boolean values) βάσει τον οποίων ορίζεται η ανάθεση ή μη κατεύθυνσης στο δίκτυο καθώς και ο συνυπολογισμός ή όχι κατά τη διαδικασία της αναζήτησης του κόστους διάσχισης μιας διαδρομής κατά την αντίθετη φορά *(reverse_cost)*, ενώ δίνει ως αποτέλεσμα τον αύξοντα αριθμό *(id)*, τον κωδικό ταυτοποίησης *(gid)* και τη γεωμετρία κάθε γεωμετρικού συνδέσμου που ανήκει στο συντομότερο μονοπάτι. Η γεωμετρία αναπαρίσταται ως μια *multilinestring* ακολουθία συντεταγμένων, καθώς επάνω στη συνάρτηση *astar_sp_directed()* και πιο συγκεκριμένα στο πεδίο της γεωμετρίας του πίνακα *'athens'* τρέχει η συνάρτηση *astext()*.

Η νέα συνάρτηση *astar_sp_directed_cost()* δέχεται ως ορίσματα το όνομα του πίνακα στον οποίο βρίσκονται καταχωρημένα τα στοιχεία του δικτύου, τους κωδικούς των κόμβων αφετηρίας και προορισμού, τις λογικές τιμές που αφορούν την κατεύθυνση του δικτύου και το αντίστροφο κόστος διάσχισης γεωμετρικού συνδέσμου, το κόστος διάσχισης της διαδρομής και το όνομα του *SEQUENCE 'resNodes'*που ταξινομεί τα αποτελέσματα. Παράλληλα, επάνω στη συνάρτηση *astar_sp_directed_cost()*, τρέχει η συνάρτηση *ST_AsKML()* η οποία μετατρέπει τη γεωμετρία σε μορφή *kml*. Τα αποτελέσματα που προκύπτουν από το τρέξιμο του αλγόριθμου A* στην περίπτωση αυτή, περιλαμβάνουν τον άυξοντα αριθμό *(id)* κάθε γεωμετρικού συνδέσμου, τμήματος της συντομότερης διαδρομής που προκύπτει, τον κωδικό ταυτοποίησης *(gid)* του γεωμετρικού συνδέσμου, το κόστος *(cost)* διάσχισης κάθε γεωμετρικού συνδέσμου, την ονομασία κάθε γεωμετρικού συνδέσμου με ελληνικούς χαρακτήρες *(greekname)*, την ονομασία κάθε γεωμετρικού συνδέσμου με λατινικούς χαρακτήρες *(latinname)* και τη γεωμετρία κάθε γεωμετρικού συνδέσμου σε μορφή *kml*. Όμοια με την περίπτωση του αλγόριθμου του Dijkstra, το κόστος διάσχισης της διαδρομής προκύπτει είτε ως η γεωμετρική απόσταση των γεωμετρικών συνδέσμων που παρεμβάλλονται μεταξύ των κόμβων αφετηρίας και προορισμού, είτε ως ο χρόνος που απαιτείται για τη διάσχιση της συγκεκριμένης διαδρομής. Ο τύπος που περιγράφει τα αποτελέσματα της συνάρτησης είναι ο *geomsc*, όπως και στην περίπτωση της συνάρτησης *dijkstra_sp_directed_cost()*. Ο κώδικας της συνάρτησης, βρίσκεται στο Παράρτημα. Στη συνέχεια, ακολουθούν δύο παραδείγματα εφαρμογής του αλγόριθμου A* για την εύρεση συντομότερης διαδρομής όπως αυτά προκύπτουν μετά την ολοκλήρωση των λειτουργιών της συνάρτησης *astar_sp_directed_cost()*.

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM
astar_sp_directed_cost('athens', 5140,4816, true, true, 'cost', 'resNodes');
```



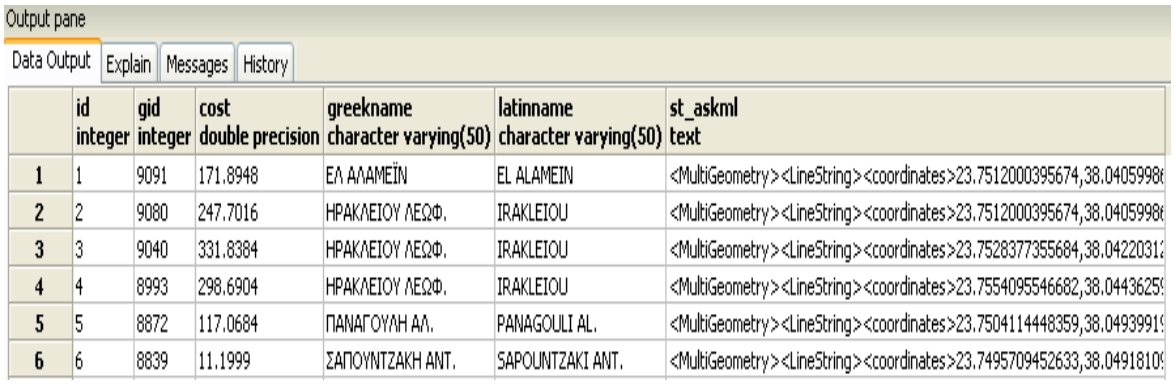
Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	8119	6.0027	ΕΘΝ. ΟΔΟΣ ΑΘΗΝΩΝ - ΘΕ	ATHINON - THESSALONIKI	<MultiGeometry><LineString><coordinates>23.8156660617865,38.1044669
2	2	8116	10.8714	ΚΟΜΒΟΣ ΕΘΝ. ΟΔΟΥ ΑΘΗ	UNK	<MultiGeometry><LineString><coordinates>23.8156660617865,38.1044669
3	3	8118	3.3098	ΑΝΩΝΥΜΟΣ	UNK	<MultiGeometry><LineString><coordinates>23.8150406346345,38.1040278
4	4	8122	41.0402	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.8157757804509,38.1023806
5	5	8126	20.4654	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.8138419586258,38.1017872
6	6	8133	53.0674	ΤΑΤΟΪΟΥ	TATOIOU	<MultiGeometry><LineString><coordinates>23.809769580719,38.09606208

Πίνακας 4-4: Ενδεικτικά Αποτελέσματα Συνάρτησης astar_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Χρόνου Διάσχισης

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM
astar_sp_directed_cost('athens', 3250,5140, true, true, 'length', 'resNodes');
```



Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	9091	171.8948	ΕΛ ΑΛΑΜΕΪΝ	EL ALAMEIN	<MultiGeometry><LineString><coordinates>23.7512000395674,38.0405998
2	2	9080	247.7016	ΗΡΑΚΛΕΙΟΥ ΛΕΩΦ.	IRAKLEIOU	<MultiGeometry><LineString><coordinates>23.7512000395674,38.0405998
3	3	9040	331.8384	ΗΡΑΚΛΕΙΟΥ ΛΕΩΦ.	IRAKLEIOU	<MultiGeometry><LineString><coordinates>23.7528377355684,38.0422031
4	4	8993	298.6904	ΗΡΑΚΛΕΙΟΥ ΛΕΩΦ.	IRAKLEIOU	<MultiGeometry><LineString><coordinates>23.7554095546682,38.0443625
5	5	8872	117.0684	ΠΑΝΑΓΟΥΛΗ ΑΛ.	PANAGOULI AL.	<MultiGeometry><LineString><coordinates>23.7504114448359,38.0493991
6	6	8839	11.1999	ΣΑΠΟΥΝΤΖΑΚΗ ΑΝΤ.	SAPOUNTZAKI ANT.	<MultiGeometry><LineString><coordinates>23.7495709452633,38.0491810

Πίνακας 4-5: Ενδεικτικά Αποτελέσματα Συνάρτησης astar_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Γεωμετρικής Απόστασης

Στα δύο παραπάνω παραδείγματα, φαίνονται τα αποτελέσματα που προκύπτουν από το τρέξιμο δύο ερωτημάτων που αφορούν τη συνάρτηση *astar_sp_directed_cost()*. Στην πρώτη περίπτωση το κόστος διάσχισης της διαδρομής προκύπτει σύμφωνα με το χρόνο που απαιτείται για τη διάσχιση κάθε γεωμετρικού συνδέσμου του συντομότερου μονοπατιού, ενώ στη δεύτερη περίπτωση σύμφωνα με τη γεωμετρική απόσταση που παρεμβάλλεται μεταξύ των κόμβων αφετηρίας και προορισμού. Η γεωμετρία των συνδέσμων αναπαρίσταται ως μια ακολουθία συντεταγμένων σε μορφή *kml*, η οποία προκύπτει ως αποτέλεσμα της συνάρτησης *ST_AsKML()* που τρέχει επάνω στην *astar_sp_directed_cost()* και μετατρέπει τη γεωμετρία σε *kml format.*

Ο αλγόριθμος Traveling Sales Person (TSP)

Ο τρίτος αλγόριθμος δρομολόγησης που έτρεξε στα πλαίσια της παρούσας εφαρμογής είναι ο αλγόριθμος Traveling Sales Person (TSP), που επιλύει το πρόβλημα του πλανόδιου πωλητή. Πρόκειται για έναν αλγόριθμο που επιλύει ένα διαφορετικής φύσεως πρόβλημα σε σχέση με τους δύο προηγούμενους αλγόριθμους, καθώς η αναζήτηση συντομότερης διαδρομής δεν αφορά δύο κόμβους αφετηρίας και προορισμού, αλλά περισσότερους κόμβους τους οποίους ορίζει ο χρήστης και συνιστούν τα σημεία από τα οποία υποχρεωτικά θα πρέπει να διέρχεται η ζητούμενη διαδρομή. Επομένως, πρόκειται για ένα πρόβλημα εύρεσης συντομότερης διαδρομής η οποία συνδέει περισσότερα των δύο εξ αρχής ορισμένων σημείων.

Η βασική συνάρτηση υλοποίησης του TSP αλγόριθμου που διαθέτει το pgRouting είναι η *tsp()*, η οποία επιλύει προσεγγιστικά το πρόβλημα με την εφαρμογή γενετικού αλγόριθμου. Ο αλγόριθμος ταξινομεί τα σημεία (κόμβους) που δίνει ο χρήστης σε μια σειρά σύμφωνα με την οποία πρέπει να προσπελαθούν, βάσει της ευκλείδιας απόστασης που υφίσταται μεταξύ τους. Η συνάρτηση δέχεται ως ορίσματα ένα ερώτημα *sql* που επιστρέφει τα *ids* των κόμβων αφετηρίας, τις *x* , *y* συντεταγμένες τους, το κόστος και το αντίστροφο κόστος, τα *ids* των κόμβων από τους οποίους θα διέρχεται το ζητούμενο μονοπάτι καθώς και τον κωδικό *(source_id)* του κόμβου εκκίνησης της ζητούμενης διαδρομής, το σημείο δηλαδή από το οποίο θα ξεκινήσει η αναζήτηση και στο οποίο θα επιστρέψει ο αλγόριθμος αφού επισκεφθεί όλα τα υπόλοιπα σημεία. Τα αποτελέσματα που δίνει η συνάρτηση *tsp* περιλαμβάνουν τους κωδικούς *(vertex_id)* των κόμβων σύμφωνα με τη σειρά που πρέπει να προσπελαθούν, τους κωδικούς *(edge_id)* των ακμών από τις οποίες διέρχεται η ζητούμενη διαδρομή και το κόστος *(cost)* που αντιστοιχεί σε

κάθε ακμή. Θα πρέπει να σημειωθεί ότι η συνάρτηση *tsp* δίνει προσεγγιστικές και όχι ακριβείς λύσεις.

Το pg Routing διαθέτει μια σειρά πρόσθετων συναρτήσεων εύρεσης συντομότερης διαδρομής οι οποίες είναι πιο σύνθετες από τη συνάρτηση *tsp()*, καθώς μέσα στην κάθε συνάρτηση τρέχουν και άλλες συναρτήσεις. Δύο από αυτές, είναι η *tsp_astar()* και η *tsp_dijkstra()* μαζί με τις αντίστοιχες *tsp_astar_directed()* και *tsp_dijkstra_directed()*. Οι συναρτήσεις *tsp_astar()* και *tsp_dijkstra()*, σε συνδυασμό με τη συνάρτηση *ST_AsKML()* που τρέχει επάνω σε αυτές, δίνουν ως αποτέλεσμα τη συντομότερη διαδρομή που συνδέει κάποια σημεία τα οποία έχει ορίσει ο χρήστης. Τα αποτελέσματα συνίστανται στους άυξοντες αριθμούς *(ids)* των γεωμετρικών συνδέσμων που περιλαμβάνονται στη συντομότερη διαδρομή, τους κωδικούς ταυτοποίησης *(gids)* των γεωμετρικών συνδέσμων καθώς και στη γεωμετρία του κάθε συνδέσμου σε μορφή *kml*. Εντός των δύο αυτών συναρτήσεων, τρέχουν οι συναρτήσεις *astar_sp_delta()* και *dijkstra_sp_delta()* αντίστοιχα. Στην περίπτωση των συναρτήσεων *tsp_astar_directed()* και *tsp_dijkstra_directed()*, τρέχουν παράλληλα οι συναρτήσεις *astar_sp_delta_directed()* και *dijkstra_sp_delta_directed()*. Οι συναρτήσεις που αναφέρθηκαν παραπάνω προϋποθέτουν τον ορισμό μιας παραμέτρου *delta*, μέσω της οποίας ορίζεται η έκταση της περιοχής αναζήτησης, προκειμένου να απλοποιηθεί η διαδικασία αναζήτησης τόσο ως προς τον απαιτούμενο χρόνο όσο και ως προς τον όγκο των δεδομένων που κάθε φορά τίθενται προς επεξεργασία.

Στα πλαίσια της εφαρμογής που σχεδιάστηκε, η συνάρτηση που τέθηκε προς επεξεργασία είναι *η tsp_dijkstra_directed()*, η οποία δέχεται ως ορίσματα το όνομα του πίνακα που περιλαμβάνει τα περιγραφικά και γεωμετρικά στοιχεία του δικτύου, τους κωδικούς *(ids)* των κόμβων από τους οποίους θα πρέπει να διέρχεται η συντομότερη διαδρομή, τον κόμβο εκκίνησης *(source)*, την παράμετρο *delta* που καθορίζει το τμήμα του δικτύου στο οποίο θα γίνει η αναζήτηση και δύο λογικές τιμές που σχετίζονται με την κατεύθυνση του δικτύου και το συνυπολογισμό του κόστους διάσχισης ενός συνδέσμου προς την αντίθετη φορά. Όπως ήδη αναφέρθηκε, η συνάρτηση αυτή τρέχει σε συνδυασμό με τη συνάρτηση *dijkstra_sp_delta_directed()* και δίνει ως αποτέλεσμα τους κωδικούς και τη γεωμετρία των ακμών από τις οποίες διέρχεται η συντομότερη διαδρομή σε μορφή *kml*.

Η συνάρτηση *tsp_dijkstra_directed()* αποτέλεσε τη βάση για τη δημιουργία μιας νέας συνάρτησης, της *tsp_sp_directed_cost()*, η δημιουργία της οποίας στηρίχθηκε στη συνάρτηση *dijkstra_sp_delta_directed_cost()*, η οποία με τη σειρά της δημιουργήθηκε στη βάση της συνάρτησης *dijkstra_sp_delta_directed()*.

Όπως και στις δύο προηγούμενες περιπτώσεις, οι απαιτήσεις της σχεδιαζόμενης εφαρμογής αφορούν στην εξαγωγή συγκεκριμένων αποτελεσμάτων, μέσω των οποίων θα καθίσταται δυνατή η λεκτική και γεωμετρική περιγραφή των αποτελεσμάτων που θα προκύπτουν από την εκάστοτε αναζήτηση βέλτιστης διαδρομής. Επομένως, είναι απαραίτητη η δημιουργία μιας νέας συνάρτησης που θα δίνει το επιθυμητό αποτέλεσμα. Όπως ήδη αναφέρθηκε σε προηγούμενη παράγραφο, η συνάρτηση *tsp_dijkstra_directed()* τρέχει σε συνδυασμό με τη συνάρτηση *dijkstra_sp_delta_directed()*. Επομένως, για την εξαγωγή του αποτελέσματος χρησιμοποιείται και ο αλγόριθμος του dijkstra για τον υπολογισμό των συντομότερων αποστάσεων μεταξύ των σημείων που έχει ορίσει ο χρήστης για τη διέλευση της ζητούμενης διαδρομής. Η διαδικασία επεξεργασίας της ήδη υπάρχουσας συνάρτησης *tsp_dijkstra_directed()* ξεκίνησε καταρχάς με τη δημιουργία μιας νέας συνάρτησης, της *dijkstra_sp_delta_directed_cost()*, η οποία τρέχει μέσα στην *tsp_sp_directed_cost()*. Η συνάρτηση *dijkstra_sp_delta_directed_cost()* σχεδιάστηκε στη βάση της *dijkstra_sp_delta_directed()*, έτσι ώστε να υπολογίζεται επιπρόσθετα το κόστος κάθε γεωμετρικού συνδέσμου που βρίσκει ο αλγόριθμος και παράλληλα οι γεωμετρικοί σύνδεσμοι να ταξινομούνται κατά μια συγκεκριμένη σειρά διαδοχής. Ο κώδικας της νέας συνάρτησης *dijkstra_sp_delta_directed_cost()* βρίσκεται στο Παράρτημα. Στη συνέχεια, σχεδιάστηκε η συνάρτηση *tsp_sp_directed_cost()*, η οποία τρέχει σε συνδυασμό με την *dijkstra_sp_delta_directed_cost()*. Η νέα συνάρτηση δέχεται ως ορίσματα το όνομα του πίνακα στον οποίο βρίσκονται αποθηκευμένα τα χωρικά στοιχεία του δικτύου, τους κωδικούς των σημείων από τα οποία θα διέρχεται η ζητούμενη συντομότερη διαδρομή, τον κωδικό του κόμβου εκκίνησης της αναζήτησης, την παράμετρο *delta* η οποία καθορίζει την έκταση της περιοχής αναζήτησης, τις λογικές τιμές που αφορούν την κατεύθυνση του δικτύου και το συνυπολογισμό του αντίστροφου κόστους διάσχισης ενός γεωμετρικού συνδέσμου, το κόστος και το όνομα *'resNodes'* της *SEQUENCE* που έχει υλοποιηθεί. Τα αποτελέσματα που προκύπτουν από το τρέξιμο της συνάρτησης είναι ο αύξων αριθμός *(id)* κάθε γεωμετρικού συνδέσμου, ο κωδικός ταυτοποίησης *(gid)* κάθε γεωμετρικού συνδέσμου, το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου *(cost)*, το όνομα κάθε γεωμετρικού συνδέσμου με ελληνικούς χαρακτήρες *(greekname)*, το όνομα κάθε γεωμετρικού συνδέσμου με λατινικούς χαρακτήρες *(latinname)* και η γεωμετρία των γεωμετρικών συνδέσμων σε μορφή *kml*. Ο κώδικας της συνάρτησης, βρίσκεται στο Παράρτημα. Στη συνέχεια, παρουσιάζονται κάποια ενδεικτικά αποτελέσματα που προέκυψαν από το τρέξιμο της συνάρτησης *tsp_sp_directed_cost()*. Στην πρώτη περίπτωση, το κόστος αφορά στη γεωμετρική απόσταση των γεωμετρικών συνδέσμων

που συνθέτουν τη συντομότερη διαδρομή που κάθε φορά προκύπτει ενώ στη δεύτερη, στο χρόνο που απαιτείται για τη διάσχιση κάθε γεωμετρικού συνδέσμου.

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM tsp_sp_directed_cost('athens', '3250,3823,3881,1013,3494', 3881, 0.1, true, true, 'length', 'resNodes');
```

Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	9021	117.9425	ΠΑΝΑΓΟΥΛΗ ΑΛ.	PANAGOULI AL.	<MultiGeometry><LineString><coordinates>23.7586811805144,38.0453878
2	2	8978	42.7417	ΠΑΝΑΓΟΥΛΗ ΑΛ.	PANAGOULI AL.	<MultiGeometry><LineString><coordinates>23.7581204011253,38.0459905
3	3	8930	171.9922	ΗΡΑΚΛΕΙΟΥ ΛΕΩΦ.	IRAKLEIOU	<MultiGeometry><LineString><coordinates>23.7581204011253,38.0459905
4	4	8911	5.2497	ΑΝΑΜΟΡΦΩΣΕΩΣ	ANAMORFOSEOS	<MultiGeometry><LineString><coordinates>23.7600781257375,38.0470716
5	5	8904	37.784	ΑΔΡΙΑΝΟΥ	ADRIANOU	<MultiGeometry><LineString><coordinates>23.7600781257375,38.0470716
6	6	8821	534.5962	ΑΤΤΙΚΗΣ	ATTIKIS	<MultiGeometry><LineString><coordinates>23.7599296295683,38.0473912

Πίνακας 4-6: Ενδεικτικά Αποτελέσματα Συνάρτησης tsp_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Γεωμετρικής Απόστασης

```
SELECT id, gid, cost, greekname, latinname, ST_AsKML(the_geom) FROM tsp_sp_directed_cost('athens', '3250,3823,3881,1013,3494', 3881, 0.1, true, true, 'cost', 'resNodes');
```

Output pane

Data Output | Explain | Messages | History

	id integer	gid integer	cost double precision	greekname character varying(50)	latinname character varying(50)	st_askml text
1	1	9021	117.9425	ΠΑΝΑΓΟΥΛΗ ΑΛ.	PANAGOULI AL.	<MultiGeometry><LineString><coordinates>23.7586811805144,38.0453878
2	2	8969	16.182	ΜΙΛΤΙΑΔΟΥ	MILTIADOU	<MultiGeometry><LineString><coordinates>23.7586811805144,38.0453878
3	3	8937	1.6556	ΜΙΛΤΙΑΔΟΥ	MILTIADOU	<MultiGeometry><LineString><coordinates>23.7601323139842,38.0464047
4	4	8923	18.4154	ΑΔΡΙΑΝΟΥ	ADRIANOU	<MultiGeometry><LineString><coordinates>23.7603313826433,38.0464097
5	5	8904	13.6022	ΑΔΡΙΑΝΟΥ	ADRIANOU	<MultiGeometry><LineString><coordinates>23.7600781257375,38.0470716
6	6	8821	64.1515	ΑΤΤΙΚΗΣ	ATTIKIS	<MultiGeometry><LineString><coordinates>23.7599296295683,38.0473912

Πίνακας 4-7: Ενδεικτικά Αποτελέσματα Συνάρτησης tsp_sp_directed_cost – Υπολογισμός Βέλτιστης Διαδρομής βάσει Χρόνου Διάσχισης

ΚΕΦΑΛΑΙΟ 5:ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ

Στο κεφάλαιο αυτό, παρουσιάζεται η διαδικασία σχεδιασμού μιας ιστοσελίδας η οποία παρέχει στο χρήστη τη δυνατότητα αναζήτησης συντομότερων διαδρομών μεταξύ σημείων που ανήκουν στο οδικό δίκτυο της Αθήνας. Η ιστοσελίδα τρέχει στο περιβάλλον του χρήστη και συνιστά το μέσο αλληλεπίδρασης του χρήστη με τη βάση δεδομένων. Ουσιαστικά, πρόκειται για το δεύτερο στάδιο σχεδιασμού της εφαρμογής δρομολόγησης που υλοποιήθηκε στα πλαίσια της παρούσας εργασίας, το οποίο περιλαμβάνει αφενός μεν το σχεδιασμό της ιστοσελίδας και τον καθορισμό των λειτουργιών της, αφετέρου δε την υλοποίηση μιας ενδιάμεσης εφαρμογής μέσω της οποίας επιτυγχάνεται η σύνδεση τη ιστοσελίδας με τη βάση δεδομένων. Στο σημείο αυτό, θα πρέπει να τονιστεί ότι το σύνολο των διαδικασιών επεξεργασίας των δεδομένων πραγματοποιείται στη βάση.

Η ιστοσελίδα αφορά το χρήστη της εφαρμογής και επιτρέπει την έμμεση επικοινωνία του με τη χωρική βάση δεδομένων. Για τη δημιουργία της, εγκαταστάθηκε ο Apache Server του MapServer. Ο σχεδιασμός της ιστοσελίδας υλοποιήθηκε σε γλώσσα HTML, ενώ ο προγραμματισμός των λειτουργιών της υλοποιήθηκε με τη βοήθεια κώδικα Javascript.

Ουσιαστικά, πρόκειται για το σχεδιασμό μιας εφαρμογής που τρέχει σε επίπεδο *client – server*, όπου ο *client* διατυπώνει μέσω της ιστοσελίδας ένα αίτημα το οποίο μεταβιβάζεται στον *server*, ακολουθεί η επεξεργασία των δεδομένων στη βάση και στη συνέχεια ο *server*, σε «συνεργασία» με τη βάση δεδομένων, επιστρέφει στον *client* ένα αποτέλεσμα. Τα βασικά συστατικά μέρη μιας τέτοιας εφαρμογής είναι ένας *server* ο οποίος λαμβάνει αιτήματα, επεξεργάζεται δεδομένα και επιστρέφει αποτελέσματα, μια ιστοσελίδα, η οποία λειτουργεί ως μέσο αλληλεπίδρασης του *client* με τον *server* υλοποιώντας κάθε φορά ορισμένες λειτουργίες και τέλος, μία «ενδιάμεση» εφαρμογή μέσω της οποίας επιτυγχάνεται η επικοινωνία της ιστοσελίδας με τον *server*.

Ο χρήστης του διαδικτυακού τόπου που δημιουργήθηκε έχει τη δυνατότητα αναζήτησης της βέλτιστης διαδρομής που δύναται να ακολουθήσει, προκειμένου να μεταβεί από ένα σημείο του δικτύου σε κάποιο άλλο, γνωρίζοντας αναλυτικά τα τμήματα

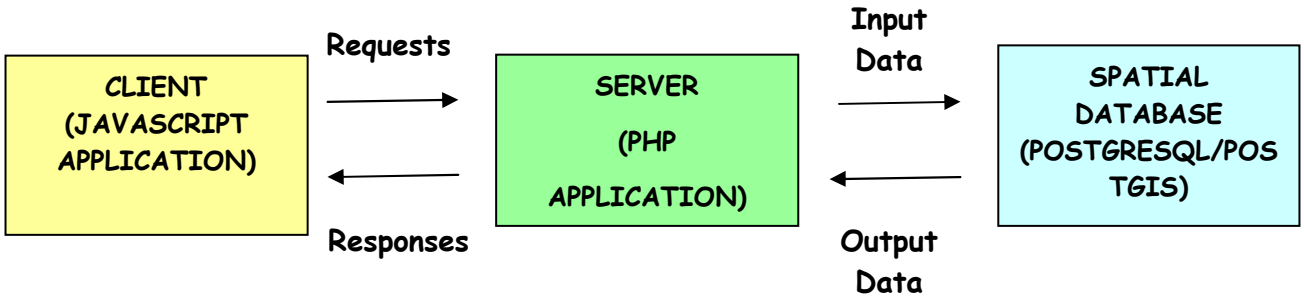
του οδικού δικτύου που πρέπει να διασχίσει, το χρόνο που απαιτείται για τη διάσχιση της διαδρομής που τον ενδιαφέρει και τη γεωμετρική απόσταση της διαδρομής. Παράλληλα, καθορίζει το τμήμα του οδικού δικτύου εντός του οποίου θα εκτελεστούν οι υπολογισμοί για την εύρεση συντομότερης διαδρομής, θέτοντας ως κριτήριο της συμπερίληψη ή όχι των τμημάτων του δικτύου που διέρχονται από το δακτύλιο, ενώ τα αποτελέσματα που λαμβάνει εμφανίζονται υπό μορφή λεκτικής περιγραφής σε έναν πίνακα και γεωμετρικής περιγραφής επάνω στο χάρτη.

Στη χωρική βάση δεδομένων, λαμβάνουν χώρα οι απαραίτητοι υπολογισμοί ενώ στην ιστοσελίδα επιλέγονται τα δεδομένα βάσει των οποίων πρόκειται να γίνει ο υπολογισμός της συντομότερης διαδρομής και εμφανίζονται τα αποτελέσματα.

Ουσιαστικά, η ιστοσελίδα συνιστά ένα διαδραστικό μέσο αλληλεπίδρασης μεταξύ του χρήστη και της χωρικής βάσης δεδομένων μέσω του javascript κώδικα που καθορίζει τις λειτουργίες της και τρέχει στον *client*. Πρόκειται λοιπόν για δύο ανεξάρτητες εφαρμογές, ένα ΣΔΧΒΔ και ένα διαδικτυακό τόπο, κάθε μια από τις οποίες επιτελεί καθορισμένες λειτουργίες. Ωστόσο, ο σκοπός για τον οποίο σχεδιάστηκε η ιστοσελίδα καθιστά αναγκαία τη δημιουργία μιας σχέσης «συνεργασίας» ανάμεσα στη βάση και τη διαδικτυακή εφαρμογή σε επίπεδο ανταλλαγής πληροφορίας. Συνεπώς, θα πρέπει να υπάρχει ένας ενδιάμεσος «συνδετικός κρίκος» ο οποίος θα συνδέει τη βάση δεδομένων με την ιστοσελίδα και τις λειτουργίες που υλοποιούνται σε αυτή και θα μεταβιβάζει πληροφορίες από τον *client* στον *server* και αντίστροφα.

Το ενδιάμεσο αυτό «εργαλείο» συνίσταται σε έναν κώδικα PHP, ο οποίος τρέχει στον *server* και μεταφέρει πληροφορία από και προς τη βάση δεδομένων και την ιστοσελίδα. Για τη σύνταξη του PHP κώδικα στα πλαίσια της παρούσας εφαρμογής, αξιοποιήθηκαν οι συναρτήσεις και οι εντολές που διαθέτει η PHP και επιτρέπουν τη σύνδεσή της με την PostgreSQL.

Στις παραγράφους που ακολουθούν, περιγράφονται αναλυτικά οι διαδικασίες σχεδιασμού της ιστοσελίδας και των λειτουργιών που εκτελούνται σε αυτή, καθώς επίσης και οι διαδικασίες ανάπτυξης του PHP κώδικα που εξασφαλίζει την επικοινωνία της ιστοσελίδας με τη βάση δεδομένων και αντίστροφα.



Σχήμα 5-1: Αρχιτεκτονική Εφαρμογής

5.1 Σχεδιασμός Ιστοσελίδας

Η διαδικασία σχεδιασμού της ιστοσελίδας αφορά ουσιαστικά στην ανάπτυξη μιας web-based εφαρμογής, η οποία θα παρέχει στο χρήστη τη δυνατότητα εύρεσης της συντομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του οδικού δικτύου, αντλώντας πληροφορίες από τη χωρική βάση δεδομένων που υλοποιήθηκε στο προηγούμενο στάδιο. Για τη δόμηση και τη μορφοποίηση της ιστοσελίδας χρησιμοποιήθηκε η γλώσσα HTML, ενώ ο καθορισμός των λειτουργιών της υλοποιήθηκε με τη βοήθεια κώδικα Javascript ο οποίος ενσωματώνεται μέσα στον HTML κώδικα. Παράλληλα, για τη χαρτογραφική απεικόνιση των αποτελεσμάτων χρησιμοποιήθηκε ένας διαδικτυακός χάρτης του google.

Η ιστοσελίδα παρέχει στο χρήστη τη δυνατότητα επιλογής των σημείων που ορίζουν τη διαδρομή του επάνω στο χάρτη, τη δυνατότητα επιλογής του αλγόριθμου βάσει του οποίου θα υπολογίζεται κάθε φορά η συντομότερη διαδρομή, τη δυνατότητα επιλογής της παραμέτρου βάσει της οποίας θα υπολογίζεται το κόστος της ζητούμενης διαδρομής καθώς και τον καθορισμό των οδικών τμημάτων που θα περιλαμβάνονται στην τελική διαδρομή που θα προκύψει με κριτήριο τη διέλευσή της από το δακτύλιο ή την κίνηση εκτός δακτυλίου. Παράλληλα, εκτός από τον πλήρη και ακριβή σχεδιασμό των λειτουργιών της ιστοσελίδας που σχετίζονται με τις χαρτογραφικές λειτουργίες και τους υπολογισμούς των αλγόριθμων δρομολόγησης, έμφαση δόθηκε στην ανάπτυξη ενός φιλικού προς το χρήστη διαδικτυακού τόπου μέσω του οποίου θα έχει τη δυνατότητα εύρεσης της διαδρομής που αναζητά εύκολα και γρήγορα.

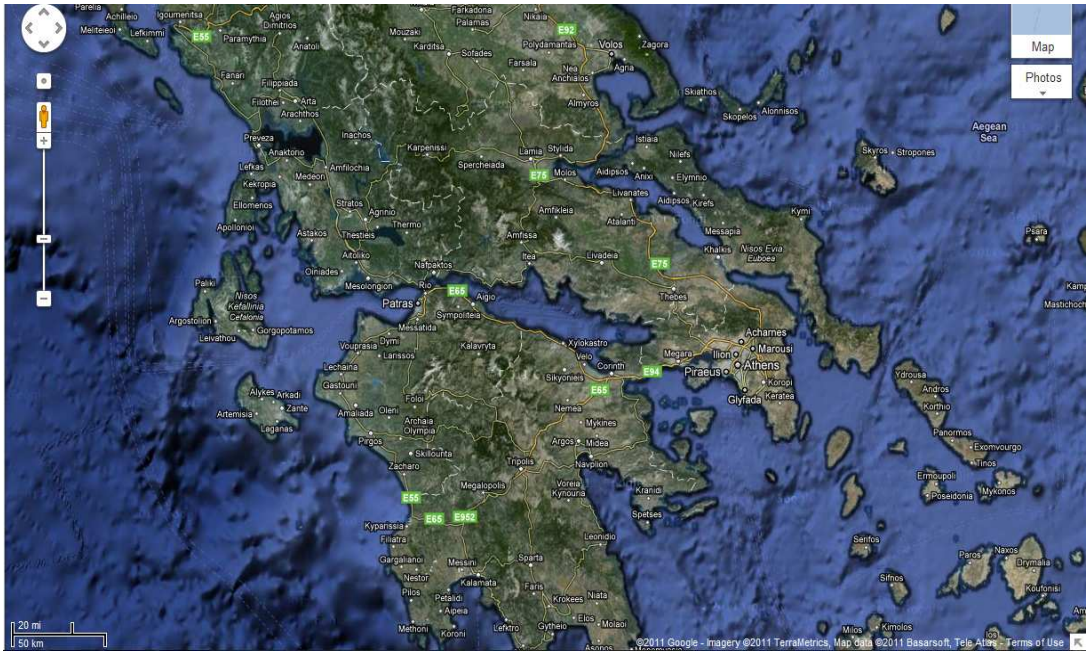
5.1.1 Το Google Maps API

Το σύνολο των λειτουργιών που εκτελούνται στην ιστοσελίδα που σχεδιάστηκε, συνδέεται είτε άμεσα είτε έμμεσα με το χαρτογραφικό υπόβαθρο το οποίο συνιστά το

βασικότερο στοιχείο της ιστοσελίδας. Η αλληλεπίδραση του χρήστη με το διαδικτυακό χάρτη είναι απαραίτητη, προκειμένου να καθοριστούν τα χωρικά δεδομένα τα οποία αποστέλλονται στη βάση όπου και λαμβάνουν χώρα οι υπολογισμοί για τον προσδιορισμό της συντομότερης διαδρομής. Παράλληλα, ο χάρτης συνιστά το μέσο οπτικοποίησης των αποτελεσμάτων που προκύπτουν από τους υπολογισμούς που λαμβάνουν χώρα στη βάση, καθώς το συντομότερο μονοπάτι που κάθε φορά προκύπτει μετά το πέρας των υπολογισμών αναπαρίσταται με τη βοήθεια ενός *kml* αρχείου, ως μια διανυσματική γραμμική οντότητα *(linestring)* στο χαρτογραφικό υπόβαθρο.

Επομένως, είναι προφανές ότι για την ανάπτυξη και τη λειτουργία της διαδικτυακής εφαρμογής, απαραίτητη προϋπόθεση είναι η δυνατότητα εκτέλεσης χαρτογραφικών λειτουργιών επάνω στο χαρτογραφικό υπόβαθρο της ιστοσελίδας. Στην προκειμένη περίπτωση, οι χαρτογραφικές λειτουργίες περιλαμβάνουν τον ορισμό σημείων επάνω στο χάρτη, τον καθορισμό λειτουργιών μεγέθυνσης/ σμίκρυνσης του χάρτη, τον καθορισμό λειτουργιών που σχετίζονται με τον ορισμό του τύπου του χαρτογραφικού υποβάθρου καθώς επίσης και λειτουργίες οπτικοποίησης των αποτελεσμάτων που κάθε φορά προκύπτουν επάνω στο χαρτογραφικό υπόβαθρο.

Προκειμένου να καταστεί εφικτή η εμφάνιση του χαρτογραφικού υποβάθρου στην ιστοσελίδα αλλά και η διεξαγωγή χαρτογραφικών λειτουργιών, αξιοποιήθηκε η υποδομή που παρέχουν τα Google Maps και το Google Maps API. Τα Google Maps συνιστούν μια διαδικτυακή υπηρεσία που παρέχεται δωρεάν από τη Google, οι χρήστες της οποίας έχουν τη δυνατότητα εύρεσης σημείων ενδιαφέροντος επάνω στο χάρτη, αναζήτησης και αναπαράστασης διαδρομών που συνδέουν ένα σημείο αφετηρίας με ένα σημείο προορισμού και επεξεργασίας των χαρτών για την αναπαράσταση διανυσματικών χωρικών δεδομένων τα οποία μοντελοποιούνται ως *kml* αρχεία. Παράλληλα, οι διαθέσιμοι χάρτες είναι δυνατό να χρησιμοποιηθούν από τους σχεδιαστές ιστοσελίδων και να ενσωματωθούν στην εκάστοτε σχεδιαζόμενη ιστοσελίδα.



Σχήμα 5-2: Υπόδειγμα Χάρτη της Google
Πηγή: [URL3]

Η Google, εκτός από τα χαρτογραφικά υπόβαθρα, παρέχει στους χρήστες μια σειρά διεπαφών, γνωστών ως Google Maps API, η αξιοποίηση των οποίων επιτρέπει την υλοποίηση ποικίλων χαρτογραφικών λειτουργιών ανάλογα με τη φύση και τις απαιτήσεις της εκάστοτε σχεδιαζόμενης εφαρμογής. Τα APIs που διαθέτει η Google είναι τα ακόλουθα [URL3]:

- **Maps Javascript API**, το οποίο επιτρέπει σε web developers την ενσωμάτωση χαρτών της Google σε ιστοσελίδες και τη διαχείριση του χάρτη και του περιεχομένου του με τη βοήθεια υλοποιημένων services.
- **Maps API for Flash**, το οποίο είναι ένα ActionScript API που επιτρέπει την ενσωμάτωση χαρτών της Google σε Flash- based ιστοσελίδες και τη διαχείριση χαρτών και του περιεχομένου τους σε τρεις διαστάσεις με τη βοήθεια υλοποιημένων services.
- **Google Earth API**, το οποίο επιτρέπει την ενσωμάτωση τρισδιάστατου χαρτογραφικού υποβάθρου σε ιστοσελίδες, όπως αυτό εμφανίζεται στο Google Earth και καθιστά δυνατή την περιήγηση σε οποιοδήποτε σημείο της γης.
- **Maps Image API**, το οποίο επιτρέπει την ταχεία ενσωμάτωση της εικόνας ενός απλού χάρτη της Google σε μια ιστοσελίδα ή μια εφαρμογή κινητού τηλεφώνου,

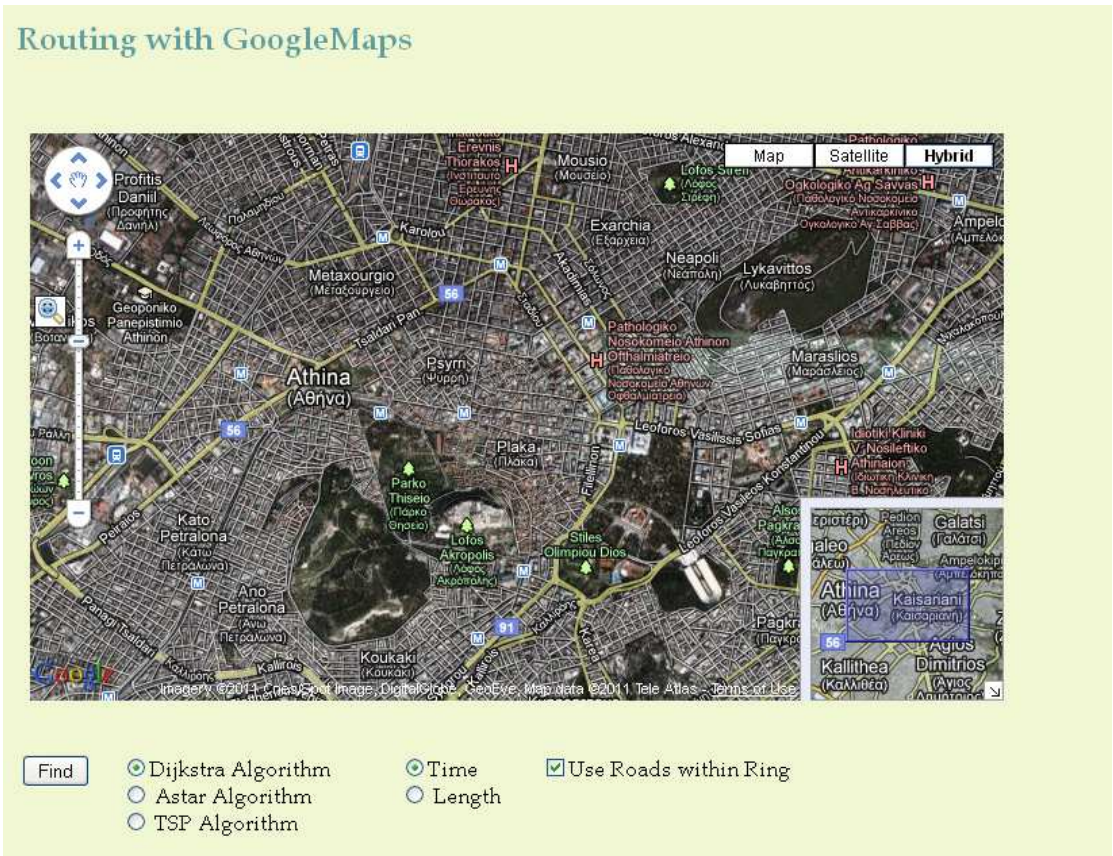
χωρίς να απαιτείται προγραμματισμός σε κώδικα Javascript ή το φόρτωμα κάποιας δυναμικής ιστοσελίδας.

- **Maps API Web Services**, το οποίο επιτρέπει μέσω της κλήσης ενός URL την πρόσβαση σε πληροφορίες γεωκωδικοποίησης, κατευθύνσεων, υψομέτρων και χαρακτηριστικών διαφόρων τοποθεσιών και τη συνακόλουθη διαχείριση των αποτελεσμάτων μέσω JSON ή XML.

Στα πλαίσια της παρούσας εργασίας, αξιοποιήθηκαν οι δυνατότητες του Maps Javascript API, προκειμένου να καθοριστούν οι λειτουργίες του χάρτη και οι διαδικασίες διαχείρισης του περιεχομένου του. Παράλληλα, αξιοποιήθηκαν οι δυνατότητες πρόσθεσης περιεχομένου στο χάρτη και οι δυνατότητες διαχείρισης συμβάντων που λαμβάνουν χώρα εντός της περιοχής που ορίζει ο χάρτης. Το Google Maps API, διαθέτει μια σειρά υλοποιημένων βιβλιοθηκών, κλάσεων και μεθόδων που απλοποιούν τη διαδικασία σύνταξης κώδικα και επιτρέπουν την αποτελεσματική διεξαγωγή ποικίλων χαρτογραφικών λειτουργιών.

5.1.2 Περιεχόμενα ιστοσελίδας

Τα περιεχόμενα της ιστοσελίδας καθορίστηκαν σύμφωνα με τις ανάγκες της σχεδιαζόμενης εφαρμογής. Η δόμηση και η μορφοποίηση των στοιχείων της υλοποιήθηκε σε κώδικα HTML. Βασικό στοιχείο της ιστοσελίδας είναι το χαρτογραφικό υπόβαθρο που χρησιμοποιείται, αφενός ως ένα διαδραστικό μέσο μεταξύ του χρήστη και της βάσης δεδομένων όπου γίνεται η επεξεργασία των δεδομένων και εξάγεται το ζητούμενο αποτέλεσμα και αφετέρου ως μέσο οπτικοποίησης των συμβάντων που λαμβάνουν χώρα στο δίκτυο. Παράλληλα, υπάρχει ένα μενού επιλογών όπου, ο χρήστης επιλέγει τον αλγόριθμο εύρεσης συντομότερης διαδρομής (Astar, Dijkstra, TSP), το κριτήριο βάσει του οποίου πρόκειται να υπολογιστεί το βάρος της ζητούμενης διαδρομής (Time, Length), εάν δηλαδή τον ενδιαφέρει η εύρεση της διαδρομής εκείνης που απαιτεί το λιγότερο χρόνο διάσχισης ή της διαδρομής που χαρακτηρίζεται από τη μικρότερη γεωμετρική απόσταση, καθώς και την επιλογή χρήσης του τμήματος του οδικού δικτύου που βρίσκεται εντός του δακτυλίου ή την εύρεση της συντομότερης διαδρομής η οποία διέρχεται εκτός του δακτυλίου (Use Roads Within Ring).



Σχήμα 5-3: Η Τελική Μορφή της Ιστοσελίδας

Η εμφάνιση των στοιχείων μιας ιστοσελίδας, υλοποιείται με τη βοήθεια κώδικα HTML. Η HTML είναι μια γλώσσα ετικετών, που αφορά αποκλειστικά τη δόμηση και τη μορφοποίηση ιστοσελίδων. Ο ορισμός των επιμέρους λειτουργιών της ιστοσελίδας απαιτεί προγραμματισμό σε άλλες γλώσσες όπως για παράδειγμα η script language, Javascript. Τα τμήματα κώδικα που αφορούν στις λειτουργίες της ιστοσελίδας εμφωλεύονται μέσα στον κώδικα HTML, συνήθως εντός των ετικετών *<HEAD></HEAD>* ή και μέσα στο κύριο σώμα του HTML κώδικα – ετικέτες *<BODY></BODY>*. Ο κώδικας HTML δομείται μέσα σε συγκεκριμένα ζεύγη ετικετών, ανάλογα με το τμήμα της ιστοσελίδας που αφορούν, ενώ υπάρχουν έτοιμες ετικέτες μέσω των οποίων καθορίζεται η μορφοποίηση του εγγράφου, όπως για παράδειγμα η στοίχιση ενός πίνακα, η εμφάνιση των επικεφαλίδων κ.λπ. Ένα έγγραφο HTML ξεκινά πάντα με την ετικέτα *<HTML>*, η οποία «πληροφορεί» τους διάφορους browsers ότι πρόκειται να φορτώσουν ένα έγγραφο HTML και κλείνει με την ετικέτα *</HTML>*, η οποία «πληροφορεί» τους browsers ότι η ανάγνωση του HTML εγγράφου ολοκληρώθηκε.

Στα πλαίσια υλοποίησης της παρούσας εφαρμογής και με τη βοήθεια HTML κώδικα, ορίστηκαν η θέση και τα χαρακτηριστικά του τίτλου της ιστοσελίδας, η θέση και το μέγεθος του διαδικτυακού χάρτη και η εισαγωγή των διαφόρων πλήκτρων και κουμπιών μέσω των οποίων ο χρήστης αλληλεπιδρά με την ιστοσελίδα, καθορίζοντας τις παραμέτρους υπολογισμού συντομότερης διαδρομής και δίνοντας την εντολή για την έναρξη των υπολογισμών στη βάση δεδομένων. Τα πλήκτρα που περιλαμβάνει η ιστοσελίδα, είναι το πληκτρο *'Find'* τύπου *'BUTTON'* η επιλογή του οποίου σημαίνει την έναρξη των υπολογισμών για την εύρεση της συντομότερης διαδρομής, τα πλήκτρα *'Time'* και *'Length'* τύπου *'radio'*, βάσει των οποίων επιλέγεται ο τρόπος υπολογισμού του βάρους της ζητούμενης διαδρομής, τα πλήκτρα επιλογής του αλγόριθμου βάσει του οποίου πρόκειται να υπολογιστεί η συντομότερη διαδρομή τα οποία είναι επίσης πλήκτρα τύπου *'radio'* και τέλος το πλήκτρο- κουτί *'Use Roads within Ring'* τύπου *'checkbox'*, όπου ο χρήστης επιλέγει έαν επιθυμεί η διαδρομή του να διέρχεται από τμήματα του οδικού δικτύου που ανήκουν στο δακτύλιο.

Για τον καθορισμό των στοιχείων της ιστοσελίδας, χρησιμοποιήθηκαν στον HTML κώδικα οι αντίστοιχες ετικέτες που ορίζουν τη δημιουργία του εκάστοτε στοιχείου, το μέγεθός του και τη θέση που πρόκειται να έχει στην ιστοσελίδα, τα χαρακτηριστικά του (μορφοποίηση) κ.λπ.

Ο κώδικας HTML που δομήθηκε στα πλαίσια της παρούσας εφαρμογής βρίσκεται στο Παράρτημα, μαζί με ορισμένες λεπτομερέστερες επεξηγήσεις. Εντός του HTML κώδικα, περιλαμβάνονται τμήματα κώδικα Javascript τα οποία καθορίζουν τις λειτουργίες που υλοποιούνται στην ιστοσελίδα και το χάρτη οι βασικότερες των οποίων περιγράφονται στο εδάφιο που ακολουθεί.

5.1.3 Λειτουργίες ιστοσελίδας

Στο εδάφιο που προηγήθηκε, έγινε η περιγραφή των διαδικασιών δόμησης και μορφοποίησης των στοιχείων που συνθέτουν την εμφάνιση της σχεδιαζόμενης ιστοσελίδας και υλοποιούνται μέσω κώδικα HTML. Παράλληλα με τη δόμηση της ιστοσελίδας, έλαβαν χώρα και ορισμένες διαδικασίες οι οποίες σχετίζονται με τις λειτουργίες που πρόκειται να υλοποιούνται στο διαδικτυακό τόπο που δημιουργήθηκε.

Οι λειτουργίες που εξυπηρετεί η ιστοσελίδα αφορούν σε διαδικασίες δρομολόγησης στο οδικό δίκτυο της Αθήνας, οι οποίες διεξάγονται με την κλήση ανάλογων συναρτήσεων που υλοποιούν αλγόριθμους δρομολόγησης. Οι επιμέρους λειτουργίες της ιστοσελίδας διακρίνονται σε λειτουργίες που διεξάγονται στο χαρτογραφικό υπόβαθρο

και αφορούν την επιλογή σημείων στο χάρτη και την οπτικοποίηση του αποτελέσματος που προκύπτει υπό μορφή *kml*, και σε λειτουργίες που αφορούν στους υπολογισμούς εύρεσης συντομότερης διαδρομής μέσω της εφαρμογής αντίστοιχων αλγόριθμων δρομολόγησης. Οι Javascript λειτουργίες τρέχουν στον *client* οι οποίες σε συνδυασμό με τις λειτουργίες (PHP) που τρέχουν στον *server*, παράγουν το ζητούμενο αποτέλεσμα, που εν προκειμένω είναι η εύρεση συνομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου.

Ο προγραμματισμός των λειτουργιών της ιστοσελίδας υλοποιήθηκε σε κώδικα Javascript, ο οποίος βρίσκεται μέσα στις ετικέτες *<HEAD></HEAD>* του HTML κώδικα. Στη συνέχεια, παρουσιάζονται οι βασικές λειτουργίες που υλοποιούνται στην ιστοσελίδα ενώ ο κώδικας της εφαρμογής βρίσκεται στο Παράρτημα.

Αρχικά, καλούνται τρία scipts τα οποία συνιστούν έτοιμα τμήματα κώδικα Javascript που χρησιμοποιούνται απευθείας από το πρόγραμμα και υλοποιούν κάποιες διαδικασίες. Τα scripts φορτώνονται πριν την ιστοσελίδα και οι αντίστοιχες διαδικασίες που υλοποιούν, λαμβάνουν χώρα όταν καλείται κάθε ένα από αυτά.

Το πρώτο script αφορά το φόρτωμα του χάρτη που περιλαμβάνεται στην ιστοσελίδα. Ο χάρτης που χρησιμοποιείται στα πλαίσια της παρούσας εφαρμογής είναι ένας χάρτης που διατίθεται από τη Google Maps. Η Google Maps μέσω της διεπαφής Google Maps API επιτρέπει τη χρήση των χαρτών της από τους χρήστες του διαδικτύου για την ανάπτυξη διαδικτυακών εφαρμογών και ιστοσελίδων. Πρόκειται για έναν κώδικα Javascript που μπορεί να καλεί ο οποιοσδήποτε χρήστης μέσα από τα δικά του προγράμματα, προκειμένου να λαμβάνει τον αντίστοιχο χάρτη που επιθυμεί. Η υπηρεσία αυτή διατίθεται δωρεάν από τη Google και το μόνο που απαιτείται από το χρήστη είναι η τήρηση κάποιων όρων που θέτει η Google για την ελεύθερη χρήση των χαρτών της καθώς και ένας λογαριασμός στη Google, ώστε να μπορεί ο χρήστης να λάβει ένα «κλειδί» που θα του επιτρέπει να χρησιμοποιεί τους Google χάρτες στην προσωπική του ιστοσελίδα.

Ο κώδικας καλεί άλλα δύο έτοιμα scripts υλοποιημένα σε κώδικα Javascript. Πρόκειται για δύο βιβλιοθήκες εκ των οποίων, η πρώτη αφορά την εκτέλεση λειτουργιών μεγέθυνσης στο χάρτη με τη βοήθεια πλαισίου (βιβλιοθήκη dragzoom.js) και η δεύτερη την εξαγωγή των αποτελεσμάτων που προκύπτουν από τους υπολογισμούς που υλοποιούνται στη βάση (βιβλιοθήκη jquery-1.6.1.js), στην ιστοσελίδα.

Ακολουθεί ο ορισμός των κλάσεων, των μεθόδων (συναρτήσεων) και των αντικειμένων με τη βοήθεια των οποίων καθορίζονται οι λειτουργίες που επιτελεί κάθε ομάδα εντολών στον κώδικα.

Η πρώτη συνάρτηση που ορίστηκε είναι η *readXML()*, η οποία διαβάζει ένα XML αρχείο που λαμβάνει κατόπιν αιτήματος, από τον *server* και ταξινομεί τα περιεχόμενά του σε έναν πίνακα που εμφανίζεται δυναμικά στην ιστοσελίδα. Το XML έγγραφο προκύπτει από τους υπολογισμούς συντομότερου μονοπατιού που υλοποιούνται στη βάση και ουσιαστικά περιλαμβάνει τη λεκτική περιγραφή της συντομότερης διαδρομής που αναζητά κάθε φορά ο χρήστης. Η συνάρτηση, έπειτα από κάθε αναζήτηση, «ανοίγει» ένα έγγραφο XML και τοποθετεί τα περιεχόμενά του στο αντίστοιχο στοιχείο *div* της ιστοσελίδας. Στην προκειμένη περίπτωση, το στοιχείο αυτό είναι ένας δυναμικός πίνακας στον οποίο καταχωρούνται: ο αύξων αριθμός *(id)* κάθε γεωμετρικού συνδέσμου που περιλαμβάνεται στο αποτέλεσμα της αναζήτησης συντομότερης διαδρομής, το όνομα κάθε γεωμετρικού συνδέσμου με ελληνικούς χαρακτήρες *(greekname)*, το όνομα κάθε γεωμετρικού συνδέσμου με λατινικούς χαρακτήρες *(latinname)* και το κόστος διάσχισης κάθε γεωμετρικού συνδέσμου *(cost)*.

Η επόμενη συνάρτηση του κώδικα είναι η *load()*, που αφορά το φόρτωμα του Google χάρτη στην ιστοσελίδα και τον ορισμό των βασικών λειτουργιών του. Η συνάρτηση περιλαμβάνει τη δημιουργία ενός νέου αντικειμένου της βασικής κλάσης του Google Maps API V2, *GMap2*, μέσω της οποίας ορίζεται η δημιουργία ενός νέου χάρτη στην HTML ιστοσελίδα καθώς και τον ορισμό των μεθόδων με τη βοήθεια των οποίων ορίζονται οι λειτουργίες που λαμβάνουν χώρα στο χάρτη. Παράλληλα, ορίζονται τα συμβάντα (events) που πρόκειται να λαμβάνουν χώρα επάνω στο χαρτογραφικό υπόβαθρο και οι αντίστοιχοι *listeners* με τη βοήθεια των οποίων ορίζονται οι ενέργειες που υλοποιούνται μέσω των συμβάντων. Τα συμβάντα συνιστούν «γεγονότα» που λαμβάνουν χώρα επάνω στο χάρτη, ύστερα από την υλοποίηση κάποιας ενέργειας από την πλευρά του χρήστη, όπως για παράδειγμα ένα κλικ του ποντικιού επάνω σε κάποιο σημείο του χάρτη. Τα συμβάντα αποτελούν επομένως ένα μέσο αλληλεπίδρασης του χρήστη με το χάρτη. Η διαχείριση των συμβάντων του Google Maps API γίνεται με τη βοήθεια συναρτήσεων, οι οποίες περιλαμβάνονται σε ένα τμήμα κώδικα που ορίζεται από την κλάση *GEventListener* και περιλαμβάνει τον ορισμό ενός *listener* που θα ανταποκρίνεται σε όποιο συμβάν πρόκειται να λαμβάνει χώρα στο χάρτη. Στα πλαίσια της σχεδιαζόμενης εφαρμογής, ορίζονται στο χάρτη δύο συμβάντα τα οποία αφορούν τις ενέργειες που πρέπει να λαμβάνουν χώρα επάνω στο χάρτη μετά από ένα δεξί ή αριστερό κλικ του χρήστη. Στην πρώτη περίπτωση, εμφανίζεται ένα *context menu* το οποίο προτρέπει το χρήστη να εισάγει στο χάρτη τα σημεία από τα οποία επιθυμεί να διέρχεται η διαδρομή του, ενώ στη δεύτερη περίπτωση το *context menu* αποκρύπτεται.

Στη συνέχεια, ακολουθούν τρεις συναρτήσεις οι οποίες ορίζουν τις λειτουργίες που πρέπει να υλοποιηθούν όταν ο χρήστης επιλέγει τις παραμέτρους βάσει των οποίων θα γίνει η αναζήτηση της συντομότερης διαδρομής. Οι συναρτήσεις αυτές είναι η *get_field()*, η οποία αφορά την επιλογή της παραμέτρου βάσει της οποίας θα υπολογιστεί το βάρος της ζητούμενης διαδρομής, η *get_method()*, η οποία αφορά τον αλγόριθμο που πρόκειται να χρησιμοποιηθεί για τον υπολογισμό της συντομότερης διαδρομής και η *get_ring()*, η οποία αφορά την επιλογή των γεωμετρικών συνδέσμων που θα ληφθούν υπόψη κατά τη διάρκεια των υπολογισμών με κριτήριο τη χωρική τους αναφορά ως προς το δακτύλιο. Οι τρεις αυτές συναρτήσεις ορίζουν ότι, οι υπολογισμοί συντομότερης διαδρομής θα υλοποιηθούν σύμφωνα με τις επιλογές που έχει κάνει ο χρήστης στα αντίστοιχα μενού της ιστοσελίδας. Επιπλέον, η συνάρτηση *get_ring()* ορίζει σε ποιόν πίνακα της βάσης θα πρέπει να γίνει η αναζήτηση της συντομότερης διαδρομής. Αυτό συμβαίνει διότι στη βάση δεδομένων, εκτός του πίνακα *'athens'* έχει δημιουργηθεί μία view, η *'no_ring'*, που περιλαμβάνει το σύνολο των οδικών τμημάτων της Αθήνας που βρίσκονται εκτός του δακτυλίου. Έτσι όταν ο χρήστης της εφαρμογής επιλέγει τη συμμετοχή στην αναζήτηση συντομότερης διαδρομής του συνόλου των οδικών αξόνων της Αθήνας, η αναζήτηση γίνεται στον πίνακα *'athens'* ενώ στην αντίθετη περίπτωση στον πίνακα *'no_ring'*.

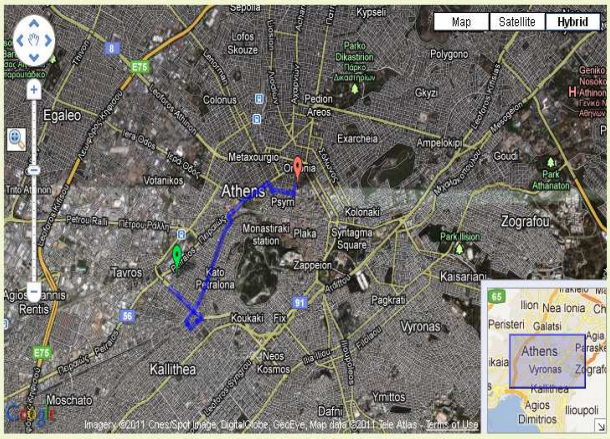
Οι συναρτήσεις *setOrigin()*, *setDestination()* και *setPoint()* που ακολουθούν, αφορούν στη σήμανση των σημείων που επιλέγει ο χρήστης στο χάρτη με *markers* και στη διαχείριση των σημείων αυτών. Ορίζονται τρεις κατηγορίες σημείων κάθε μια από τις οποίες σημαίνεται με διαφορετικό τρόπο και επιτελεί συγκεκριμένες λειτουργίες. Η πρώτη συνάρτηση, ορίζει την εισαγωγή ενός πράσινου *marker* για τη σήμανση ενός σημείου αφετηρίας, η δεύτερη την εισαγωγή ενός κόκκινου *marker* για τη σήμανση ενός σημείου προορισμού και η τρίτη την εισαγωγή ενός μπλε *marker* στην περίπτωση που ο χρήστης επιθυμεί να ορίσει περισσότερα των δύο σημείων από τα οποία θα διέρχεται η διαδρομή του. Ο ορισμός των *markers* γίνεται με τη βοήθεια της κλάσης *GIcon* η οποία δημιουργεί ένα νέο αντικείμενο που αναπαριστά το σημείο που κάθε φορά επιλέγεται. Παράλληλα, ορίζονται νέα συμβάντα που αφορούν τη διαγραφή των σημείων που επιλέγει ο χρήστης με κλικ επάνω σε κάθε *marker*.

Το τελευταίο τμήμα του κώδικα περιλαμβάνει τον ορισμό της συνάρτησης *calculate()*, με τη βοήθεια της οποίας εκτελούνται οι απαραίτητοι υπολογισμοί εύρεσης συντομότερης διαδρομής και εμφανίζονται τα αντίστοιχα αποτελέσματα στην ιστοσελίδα. Σύμφωνα με το *input* που ορίζεται κάθε φορά από το χρήστη και αφορά τις συντεταγμένες των σημείων που επιλέγει στο χάρτη και τις παραμέτρους βάσει των οποίων θα υπολογιστεί η συντομότερη διαδρομή, η συνάρτηση *calculate()* στέλνει ένα

αίτημα στον *server* για την επεξεργασία των δεδομένων εισόδου και τον υπολογισμό του αποτελέσματος που προκύπτει. Αφού ολοκληρωθούν στη βάση οι απαραίτητοι υπολογισμοί, το εκάστοτε αποτέλεσμα αποστέλλεται μέσω του *server* στην ιστοσελίδα όπου και παρουσιάζεται ως λεκτική περιγραφή σε ένα δυναμικό πίνακα και ως πολυγραμμή επάνω στο χαρτογραφικό υπόβαθρο.

Στη συνέχεια, παρουσιάζονται κάποια ενδεικτικά αποτελέσματα που προέκυψαν ύστερα από την αναζήτηση της συντομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου με τη βοήθεια των αλγόριθμων Dijkstra, A* και TSP αντίστοιχα. Ο κώδικας της εφαρμογής βρίσκεται στο Παράρτημα, μαζί με τα απαραίτητα σχόλια που αποσαφηνίζουν το περιεχόμενό του.

Routing with GoogleMaps



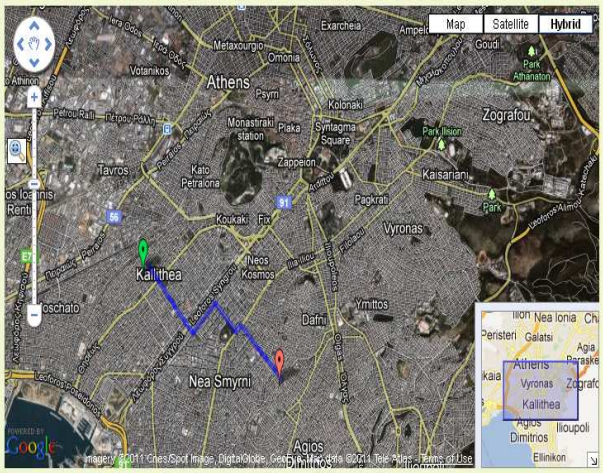
Find
Dijkstra Algorithm
Astar Algorithm
TSP Algorithm
Time
Length
Use Roads within Ring

id	greekname	latinname	cost
1	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	15.6134
2	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	45.9064
3	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	55.2901
4	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	21.0413
5	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	18.7421
6	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	5.6431
7	ΤΣΑΛΔΑΡΗ ΠΑΝ.	TSALDARI PAN.	36.0183
8	ΑΝΩΝΥΜΟΣ	UNK	8.8029
9	ΚΡΕΜΟΥ Γ.	KREMOU G.	65.1032
10	ΧΑΡΟΚΟΠΟΥ	CHAROKOPOU	10.4989
11	ΧΑΡΟΚΟΠΟΥ	CHAROKOPOU	0.9066
12	ΑΝΩΝΥΜΟΣ	UNK	4.1051
13	ΘΕΣΣΑΛΟΝΙΚΗΣ	THESSALONIKIS	15.3572
14	ΘΕΣΣΑΛΟΝΙΚΗΣ	THESSALONIKIS	11.3502
15	ΘΕΣΣΑΛΟΝΙΚΗΣ	THESSALONIKIS	17.9673
16	ΘΕΣΣΑΛΟΝΙΚΗΣ	THESSALONIKIS	17.1955
17	ΘΕΣΣΑΛΟΝΙΚΗΣ	THESSALONIKIS	92.725
18	ΠΕΡΣΕΦΟΝΗΣ	PERSEFONIS	23.7622
19	ΠΕΡΣΕΦΟΝΗΣ	PERSEFONIS	37.4956
20	ΠΕΙΡΑΙΩΣ	PEIRAIOS	19.2339
21	ΠΕΙΡΑΙΩΣ	PEIRAIOS	37.844
22	ΠΕΙΡΑΙΩΣ	PEIRAIOS	31.5991
23	ΠΕΙΡΑΙΩΣ	PEIRAIOS	23.5839
24	ΠΕΙΡΑΙΩΣ	PEIRAIOS	16.4469

Σχήμα 5-4: Παράδειγμα Δρομολόγησης με τον Αλγόριθμο του Dijkstra

Στο σχήμα 5-4, παρουσιάζεται ένα παράδειγμα εύρεσης συντομότερης διαδρομής από κάποιο σημείο αφετηρίας που βρίσκεται στα Κάτω Πετράλωνα, προς ένα σημείο προορισμού που βρίσκεται στην περιοχή του Ψυρρή. Η διαδρομή υπολογίζεται με τη βοήθεια του αλγόριθμου του Dijkstra, το κόστος της διαδρομής υπολογίζεται βάσει του χρόνου που απαιτείται για τη μετάβαση από το σημείο αφετηρίας στο σημείο προορισμού ενώ στην αναζήτηση συμπεριλαμβάνονται τα οδικά τμήματα του δικτύου που ανήκουν στο δακτύλιο. Τα αποτελέσματα της αναζήτησης εμφανίζονται οπτικοποιημένα στο χάρτη ως μια διανυσματική γραμμική οντότητα (αρχείο kml) και ως λεκτική περιγραφή στο δυναμικό πίνακα της ιστοσελίδας, ο οποίος περιλαμβάνει τους γεωμετρικούς συνδέσμους που συναποτελούν τη ζητούμενη διαδρομή και το κόστος που απαιτείται για τη διάσχιση καθενός από αυτούς.

Routing with GoogleMaps



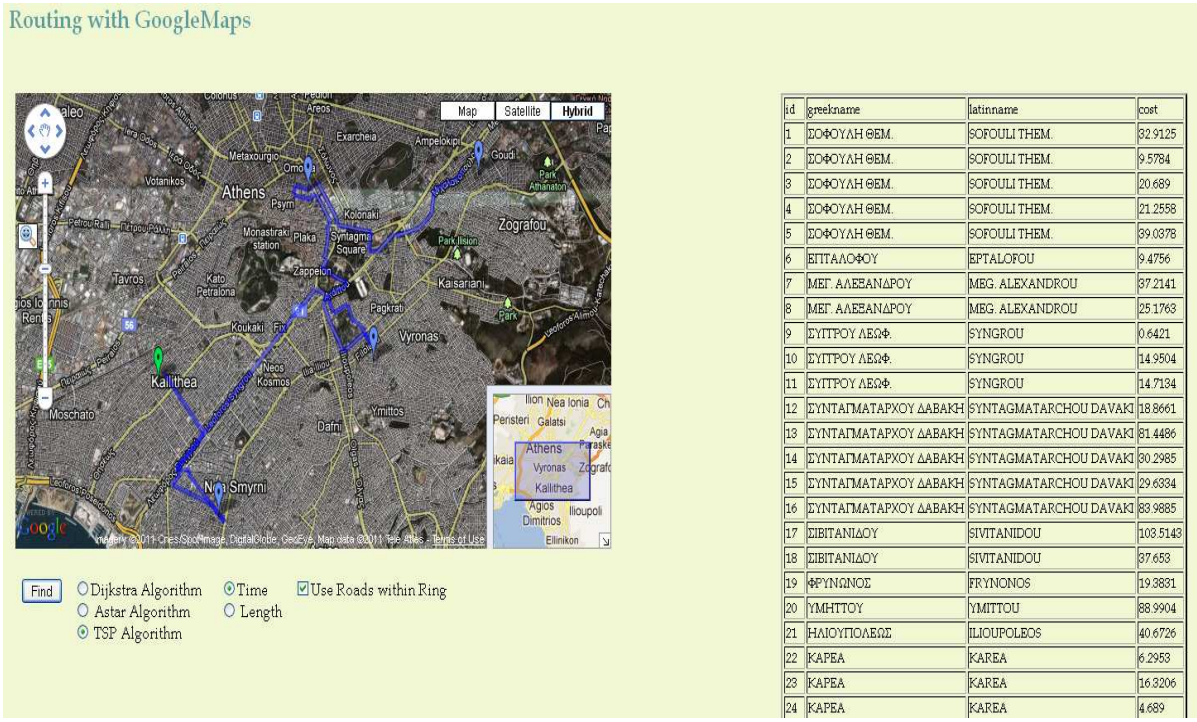
id	greekname	latinname	cost
1	ΣΙΒΙΤΑΝΙΔΟΥ	SIVITANIDOU	43.9285
2	ΣΙΒΙΤΑΝΙΔΟΥ	SIVITANIDOU	75.2831
3	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	90.9875
4	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	20.5154
5	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	87.8731
6	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	61.0865
7	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	17.2256
8	ΑΓ. ΦΩΤΕΙΝΗΣ	AG. FOTEINIS	107.4631
9	ΠΛΑΣΤΗΡΑ Ν.	PLASTIRA N.	3.3415
10	ΠΛΑΣΤΗΡΑ Ν.	PLASTIRA N.	132.3176
11	ΕΦΕΣΟΥ	EFESOU	548.9489
12	ΒΕΝΙΖΕΛΟΥ ΕΛΕΥΘ. ΛΕΩΦ.	VENIZELOU ELEFTH.	14.3254
13	ΑΙΓΑΙΟΥ	AIGAIOU	441.1288
14	ΑΙΓΑΙΟΥ	AIGAIOU	88.4124
15	ΑΙΓΑΙΟΥ	AIGAIOU	113.1871
16	ΑΙΓΑΙΟΥ	AIGAIOU	209.2462
17	ΑΙΓΑΙΟΥ	AIGAIOU	130.2731
18	ΑΙΓΑΙΟΥ	AIGAIOU	60.2943

Find ○ Dijkstra Algorithm ○ Astar Algorithm ○ TSP Algorithm ○ Time ○ Length ☑ Use Roads within Ring

*Σχήμα 5-5: Παράδειγμα Δρομολόγησης με τον Αλγόριθμο A**

Στο σχήμα 5-5, παρουσιάζεται ένα παράδειγμα εύρεσης συντομότερης διαδρομής από κάποιο σημείο αφετηρίας που βρίσκεται στην Καλλιθέα, προς ένα σημείο προορισμού που βρίσκεται στην περιοχή της Νέας Σμύρνης. Η διαδρομή υπολογίζεται με τη βοήθεια του αλγόριθμου A*, το κόστος της διαδρομής υπολογίζεται βάσει της γεωμετρικής απόστασης που απαιτείται να διανυθεί, για τη μετάβαση από το σημείο αφετηρίας στο σημείο προορισμού, ενώ στην αναζήτηση συμπεριλαμβάνονται τα οδικά τμήματα του

δικτύου που ανήκουν στο δακτύλιο. Τα αποτελέσματα της αναζήτησης εμφανίζονται οπτικοποιημένα στο χάρτη ως μια διανυσματική γραμμική οντότητα (αρχείο kml) και ως λεκτική περιγραφή στο δυναμικό πίνακα της ιστοσελίδας, ο οποίος περιλαμβάνει τους γεωμετρικούς συνδέσμους που συναποτελούν τη ζητούμενη διαδρομή και το κόστος που απαιτείται για τη διάσχιση καθενός από αυτούς.



id	greekname	latinname	cost
1	ΣΟΦΟΥΛΗ ΘΕΜ.	SOFOULI THEM.	82.9125
2	ΣΟΦΟΥΛΗ ΘΕΜ.	SOFOULI THEM.	9.5784
3	ΣΟΦΟΥΛΗ ΘΕΜ.	SOFOULI THEM.	20.689
4	ΣΟΦΟΥΛΗ ΘΕΜ.	SOFOULI THEM.	21.2558
5	ΣΟΦΟΥΛΗ ΘΕΜ.	SOFOULI THEM.	89.0378
6	ΕΠΤΑΛΟΦΟΥ	EPTALOFOU	9.4756
7	ΜΕΓ. ΑΛΕΞΑΝΔΡΟΥ	MEG. ALEXANDROU	87.2141
8	ΜΕΓ. ΑΛΕΞΑΝΔΡΟΥ	MEG. ALEXANDROU	25.1763
9	ΣΥΓΓΡΟΥ ΛΕΩΦ.	SYNGROU	0.6421
10	ΣΥΓΓΡΟΥ ΛΕΩΦ.	SYNGROU	14.9504
11	ΣΥΓΓΡΟΥ ΛΕΩΦ.	SYNGROU	14.7134
12	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	18.8661
13	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	81.4486
14	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	80.2985
15	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	29.6334
16	ΣΥΝΤΑΓΜΑΤΑΡΧΟΥ ΔΑΒΑΚΗ	SYNTAGMATARCHOU DAVAKI	83.9885
17	ΣΙΒΙΤΑΝΙΔΟΥ	SIVITANIDOU	108.5143
18	ΣΙΒΙΤΑΝΙΔΟΥ	SIVITANIDOU	87.653
19	ΦΡΥΝΩΝΟΣ	FRYNONOS	19.3831
20	ΥΜΗΤΤΟΥ	YMITTOU	88.9904
21	ΗΛΙΟΥΠΟΛΕΩΣ	ILIOUPOLEOS	40.6726
22	ΚΑΡΕΑ	KAREA	6.2953
23	ΚΑΡΕΑ	KAREA	16.3206
24	ΚΑΡΕΑ	KAREA	4.689

Σχήμα 5-6: Παράδειγμα Δρομολόγησης με τον Αλγόριθμο TSP

Στο σχήμα 5-6, παρουσιάζεται ένα παράδειγμα εύρεσης συντομότερης διαδρομής μεταξύ πέντε σημείων του δικτύου. Η διαδρομή υπολογίζεται με τη βοήθεια του αλγόριθμου TSP, το κόστος της διαδρομής υπολογίζεται βάσει του χρόνου που απαιτείται για τη διαδοχική επίσκεψη των πέντε παραπάνω σημείων ενώ στην αναζήτηση συμπεριλαμβάνονται τα οδικά τμήματα του δικτύου που ανήκουν στο δακτύλιο. Τα αποτελέσματα της αναζήτησης εμφανίζονται οπτικοποιημένα στο χάρτη ως μια διανυσματική γραμμική οντότητα (αρχείο kml) και ως λεκτική περιγραφή στο δυναμικό πίνακα της ιστοσελίδας, ο οποίος περιλαμβάνει τους γεωμετρικούς συνδέσμους που συναποτελούν τη ζητούμενη διαδρομή και το κόστος που απαιτείται για τη διάσχιση καθενός από αυτούς.

Στην περίπτωση αυτή, η αναζήτηση συντομότερης διαδρομής γίνεται με τη βοήθεια του αλγόριθμου Traveling Sales Person, με τη βοήθεια του οποίου επιτυγχάνεται η προσεγγιστική επίλυση του προβλήματος του πλανόδιου πωλητή. Στα πλαίσια της παρούσας εφαρμογής, ο χρήστης έχει τη δυνατότητα επιλογής των σημείων του δικτύου που επιθυμεί να επισκεφθεί και επιλέγοντας ως μέθοδο αναζήτησης τον αλγόριθμο TSP λαμβάνει το αντίστοιχο αποτέλεσμα βάσει του οποίου ορίζεται η σειρά επισκεψιμότητας των επιλεγμένων σημείων, ούτως ώστε να πληρούνται τα κριτήρια που κάθε φορά ορίζονται για την αναζήτηση της βέλτιστης διαδρομής. Ο χρήστης έχει τη δυνατότητα επιλογής του σημείου από το οποίο επιθυμεί να ξεκινήσει η αναζήτηση. Ωστόσο, εάν η διαδρομή που προκύπτει δεν είναι η βέλτιστη, ο αλγόριθμος επιλέγει διαφορετικό σημείο αφετηρίας και δίνει ως αποτέλεσμα τη συντομότερη διαδρομή που κάθε φορά προκύπτει.

5.2 Σύνδεση της Διαδικτυακής Εφαρμογής με τη Βάση Δεδομένων

Στα εδάφια που προηγήθηκαν, παρουσιάστηκαν οι διαδικασίες υλοποίησης μιας διαδικτυακής εφαρμογής η οποία παρέχει στο χρήστη υπηρεσίες δρομολόγησης. Ο χρήστης του διαδικτυακού τόπου, έχει τη δυνατότητα αναζήτησης της βέλτιστης διαδρομής που δύναται να ακολουθήσει προκειμένου να μεταβεί από ένα σημείο του δικτύου σε κάποιο άλλο, γνωρίζοντας αναλυτικά τα τμήματα του οδικού δικτύου που πρέπει να διασχίσει, το χρόνο που απαιτείται για τη διάσχιση της διαδρομής που τον ενδιαφέρει και τη γεωμετρική απόσταση της διαδρομής. Παράλληλα, έχει τη δυνατότητα επιλογής του αλγόριθμου βάσει του οποίου υπολογίζεται κάθε φορά η ζητούμενη διαδρομή, της παραμέτρου βάσει της οποίας υπολογίζεται το κόστος της ζητούμενης διαδρομής καθώς επίσης και των οδικών τμημάτων που επιθυμεί να συμμετάσχουν στη διαδικασία της αναζήτησης με κριτήριο τη διέλευσή τους από το δακτύλιο.

Στο παρόν εδάφιο, περιγράφονται οι διαδικασίες σχεδιασμού μιας ενδιάμεσης εφαρμογής η οποία τρέχει στον *server* και καθιστά δυνατή της σύνδεση της ιστοσελίδας που τρέχει στο περιβάλλον του *client* με τη βάση δεδομένων όπου λαμβάνουν χώρα οι απαραίτητοι για το υπολογισμό της βέλτιστης διαδρομής, υπολογισμοί. Η εφαρμογή αυτή, υλοποιήθηκε σε κώδικα PHP και συνιστά το «συνδετικό κρίκο» μεταξύ της ιστοσελίδας και της βάσης δεδομένων.

5.2.1 Η PHP και η PostgreSQL

Η PHP χαρακτηρίζεται ως μια 'scripting' γλώσσα προγραμματισμού καθώς όπως και η Javascript μπορεί να χρησιμοποιηθεί για τη δημιουργία τμημάτων κώδικα (scripts) τα οποία εμφωλεύονται μέσα σε HTML έγγραφα. Συνιστά μια γλώσσα προγραμματισμού για τη δημιουργία ιστοσελίδων με δυναμικό περιεχόμενο το οποίο παράγεται σε δυναμικό χρόνο. Ένας κώδικας PHP, τρέχει σε κάποιο *server* και το περιεχόμενό του δεν είναι προσβάσιμο από τους χρήστες της ιστοσελίδας. Στην περίπτωση που ορισμένα τμήματα του κώδικα περιλαμβάνουν πληροφορία που πρέπει να παρέχεται στο χρήστη, ο *server* κατά τη διάρκεια επεξεργασίας του κώδικα «βλέπει» τα τμήματα εκείνα που πρέπει να είναι ορατά, τα μετατρέπει σε HTML μορφή και τα επιστρέφει στο περιβάλλον του *client* [URL10], [URL11].

Τα βασικά πλεονεκτήματα που εμφανίζει η PHP ως προγραμματιστικό εργαλείο συνίστανται στα ακόλουθα [URL10], [URL11]:

- Συμβάλλει στη μείωση του απαιτούμενου χρόνου δημιουργίας μεγάλων websites.
- Παρέχει στο δημιουργό της διαδικτυακής εφαρμογής τη δυνατότητα συλλογής πληροφοριών σχετικών με τους χρήστες της εφαρμογής.
- Καθιστά διαθέσιμα πολλά online εργαλεία και διαθέτει ένα πλήθος έτοιμων συναρτήσεων και εντολών.
- Ενσωματώνεται εύκολα μέσα σε HTML έγγραφο σε σχέση με άλλες γλώσσες.
- Υποστηρίζει πολλά ΣΔΒΔ και ως εκ τούτου επιτρέπει τη σύνδεση και την άντληση πληροφοριών από μια βάση δεδομένων με χρήση προκαθορισμένων συναρτήσεων.

Η σύνταξη της PHP μοιάζει με την αντίστοιχη της C ή της Java, ενώ ο PHP κώδικας ενσωματώνεται σε μια ετικέτα η οποία περιλαμβάνει ένα λατινικό ερωτηματικό *<?php …….. ?>*.

Στα πλαίσια ανάπτυξης της παρούσας εφαρμογής δρομολόγησης, αξιοποιήθηκε η δυνατότητα σύνδεσης PHP κώδικα με μια βάση δεδομένων για τη λήψη πληροφοριών. Συντάχθηκε κώδικας PHP ο οποίος τρέχει στον *server*, λαμβάνει τα δεδομένα που εισάγει στην ιστοσελίδα ο χρήστης και επιστρέφει τα αποτελέσματα που προκύπτουν από την επεξεργασία των δεδομένων εισαγωγής στη βάση. Η σύνδεση του PHP κώδικα με την PostgreSQL υλοποιήθηκε με τη βοήθεια ειδικών συναρτήσεων που διαθέτει η PHP και

αφορούν τη σύνδεση με τη βάση δεδομένων και τη μεταβίβαση των λειτουργιών που πρέπει να υλοποιηθούν στη βάση, προκειμένου να προκύψει το ζητούμενο αποτέλεσμα.

Η λήψη δεδομένων από την PostgreSQL, μέσω κώδικα PHP, συνιστά μια διαδικασία που ολοκληρώνεται μέσα από την εκτέλεση μιας σειράς επιμέρους διαδοχικών βημάτων. Το πρώτο βήμα αφορά στον ορισμό των απαιτούμενων πληροφοριών μέσα στον PHP κώδικα, οι οποίες είναι απαραίτητες προκειμένου να πραγματοποιηθεί η σύνδεση του PHP κώδικα με την PostgreSQL. Οι πληροφορίες αυτές αφορούν στο όνομα του *server* της PostgreSQL, το *username* και το *password* που απαιτείται για τη σύνδεση με τον *server* της PostgreSQL και το όνομα της βάσης δεδομένων με την οποία πρόκειται να συνδεθεί ο κώδικας PHP. Τα παραπάνω στοιχεία συνιστούν τιμές, οι οποίες εκχωρούνται στις αντίστοιχες μεταβλητές του κώδικα PHP. Η σύνδεση του PHP κώδικα με την PostgreSQL υλοποιείται με τη βοήθεια της συνάρτησης *pg_connect()*, η οποία δέχεται ως ορίσματα τα στοιχεία που αφορούν τη σύνδεση με την αντίστοιχη βάση δεδομένων, δηλαδή τα ονόματα του *server* και της βάσης καθώς και το *username* και το *password* μέσω των οποίων επιτυγχάνεται στο περιβάλλον του pgAdmin η σύνδεση με τη βάση. Η συνάρτηση *pg_connect()* επιστρέφει ένα *link identifier* το οποίο αποθηκεύεται στη μεταβλητή στην οποία έχει εκχωρηθεί η συνάρτηση *pg_connect()*. Ο *identifier* αυτός χρησιμοποιείται μέσα στο PHP script, κάθε φορά που ο PHP κώδικας πρέπει να επικοινωνήσει με τη βάση [23].

Μέσω του PHP κώδικα και αφού έχει επιτευχθεί η σύνδεση με τη βάση δεδομένων, είναι δυνατή η αποστολή ερωτημάτων (queries) στη βάση για τη λήψη της ανάλογης πληροφορίας από αυτή. Η συνάρτηση μέσω της οποίας ένας PHP κώδικας είναι δυνατό να αποστείλει ένα *query* στη βάση δεδομένων είναι η *pg_query()*, η οποία δέχεται ως ορίσματα τη μεταβλητή στην οποία έχει εκχωρηθεί η τιμή που επιστρέφεται όταν επιτυγχάνεται η σύνδεση του PHP κώδικα με τη βάση δεδομένων και τη μεταβλητή στην οποία έχει εκχωρηθεί το SQL ερώτημα που ο PHP κώδικας πρέπει να αποστείλει στη βάση. Το αποτέλεσμα που επιστρέφει η βάση μετά την επεξεργασία ενός SQL ερωτήματος εκχωρείται στη μεταβλητή που αντιστοιχεί στη συνάρτηση *pg_query()* [23].

Μετά την ανάκτηση των απαραίτητων πληροφοριών από τη βάση και για λόγους εξοικονόμησης χώρου μνήμης, ο PHP κώδικας αποσυνδέεται από τη βάση δεδομένων με τη βοήθεια της συνάρτησης *pg_close()* η οποία δέχεται ως όρισμα τη μεταβλητή στην οποία είχε εκχωρηθεί η τιμή που επιστράφηκε όταν έγινε η σύνδεση του PHP κώδικα με τη βάση δεδομένων [23].

Η PHP διαθέτει μια σειρά έτοιμων συναρτήσεων οι οποίες χρησιμοποιούνται για την εκτέλεση διάφορων λειτουργιών στη βάση δεδομένων. Η συνάρτηση *pg_num_rows()* για

παράδειγμα χρησιμοποιείται, για την καταμέτρηση του αριθμού των εγγραφών του πίνακα που περιλαμβάνει τα αποτελέσματα που προέκυψαν μετά την εκτέλεση ενός SQL ερωτήματος. Η συνάρτηση *pg_fetch_row()* επιστρέφει τις στήλες που περιλαμβάνει μια σειρά του πίνακα του αποτελέσματος ως έναν πίνακα (array) που περιλαμβάνει το σύνολο των στοιχείων που αφορούν μια εγγραφή [23].

Στο εδάφιο που ακολουθεί, παρουσιάζονται οι βασικές λειτουργίες του κώδικα PHP που δημιουργήθηκε στα πλαίσια της παρούσας εφαρμογής για την υλοποίηση των απαραίτητων υπολογισμών στη βάση δεδομένων.

5.2.2 Υλοποίηση κώδικα PHP

Ο κώδικας PHP που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας, συνιστά μια «γέφυρα επικοινωνίας» μεταξύ της ιστοσελίδας και της βάσης δεδομένων. Οι λειτουργίες που υλοποιεί περιλαμβάνουν τη διατύπωση ερωτημάτων στη βάση και τη λήψη των αντίστοιχων αποτελεσμάτων από αυτή, καθώς και τη μεταφορά τους στην ιστοσελίδα όπου και τίθενται στη διάθεση του *client*.

Στις παραγράφους που ακολουθούν περιγράφονται οι λειτουργίες που επιτελούνται στα βασικά σημεία του PHP κώδικα καθώς και ο τρόπος με τον οποίο επιτυγχάνεται, με τη συμβολή του PHP κώδικα, η επικοινωνία ανάμεσα στον *client* και τον *server*.

Αρχικά, ορίζεται ένα στιγμιότυπο της κλάσης *DOMDocument()* το οποίο χρησιμοποιείται για την ανάλυση του XML εγγράφου όπου πρόκειται να καταχωρούνται τα αποτελέσματα που επιστρέφει η βάση δεδομένων, κάθε φορά που απαντά σε ένα SQL ερώτημα μέσω του PHP κώδικα. Στη συνέχεια, ακολουθούν οι μέθοδοι που ορίζουν τη δημιουργία του XML εγγράφου και την καταχώρηση του εκάστοτε επιστρεφόμενου αποτελέσματος σε αυτό. Ακολουθούν οι αντίστοιχες εντολές για τη δημιουργία του KML αρχείου, το οποίο πρόκειται να περιλαμβάνει τη γεωμετρική περιγραφή των αποτελεσμάτων. Ορίζεται ένα νέο στιγμιότυπο της κλάσης *DOMDocument()* και στη συνέχεια ακολουθούν οι μέθοδοι βάσει των οποίων υλοποιείται η δημιουργία ενός KML αρχείου και η καταχώρηση των γεωμετρικών στοιχείων σε αυτό.

Στη συνέχεια, ορίζονται οι μεταβλητές στις οποίες καταχωρούνται οι παραμετρικές τιμές που λαμβάνουν τα ορίσματα των συναρτήσεων εύρεσης συντομότερης διαδρομής, τιμές που ορίζονται στο περιβάλλον της ιστοσελίδας από το χρήστη. Οι παραμετρικές τιμές αφορούν τις συντεταγμένες των σημείων αφετηρίας και προορισμού, τη μέθοδο βάσει της οποίας θα υπολογιστεί η συντομότερη διαδρομή, το πεδίο που αφορά το κριτήριο βάσει του οποίου θα υπολογιστεί το κόστος της διαδρομής, καθώς και ο πίνακας

που πρόκειται να χρησιμοποιηθεί για τον υπολογισμό της συντομότερης διαδρομής ανάλογα με το συνυπολογισμό ή όχι στην αναζήτηση των οδικών τμημάτων που ανήκουν στο δακτύλιο.

Με τη βοήθεια της συνάρτησης *pg_connect()* ξεκινά η διαδικασία σύνδεσης του PHP κώδικα με τη βάση, ενώ στη συνέχεια ορίζονται τα δύο SQL ερωτήματα όπου ο PHP κώδικας αποστέλλει στη βάση. Ο PHP κώδικας λαμβάνει από την ιστοσελίδα κάποια δεδομένα εισόδου, τα οποία στη συνέχεια και αποστέλλει στη βάση, μαζί με τα SQL ερωτήματα που τίθενται για την επεξεργασία τους και την εξαγωγή του τελικού αποτελέσματος. Τα δεδομένα που λαμβάνει ο κώδικας αφορούν καταρχάς, τα δύο σημεία αφετηρίας και προορισμού που επιλέγει ο χρήστης επάνω στο χάρτη και πιο συγκεκριμένα τις συντεταγμένες των σημείων *x1*, *y1*, και *x2*, *y2*. Στο σημείο αυτό, ανακύπτει ένα ζήτημα το οποίο εντοπίζεται στο γεγονός ότι, οι συντεταγμένες των σημείων που κάθε φορά επιλέγει ο χρήστης δεν είναι απαραίτητα αποθηκευμένες στη βάση. Η λύση στο πρόβλημα αυτό ανάγεται στην εύρεση των πλησιέστερων στα σημεία αφετηρίας και προορισμού σημείων, οι συντεταγμένες των οποίων βρίσκονται αποθηκευμένες στη βάση. Για το λόγο αυτό, δημιουργήθηκε μια νέα όψη (view) στη βάση, η οποία περιλαμβάνει τους κωδικούς και τις συντεταγμένες των κόμβων του δικτύου. Ακολούθως, στον PHP κώδικα διατυπώνεται ένα SQL ερώτημα για την εύρεση των σημείων που βρίσκονται εντός μιας συγκεκριμένης ελάχιστης απόστασης από τα σημεία αφετηρίας και προορισμού που επιλέγει κάθε φορά ο χρήστης. Οι συντεταγμένες των δύο πλησιέστερων σημείων, εκχωρούνται σε τέσσερις μεταβλητές που έχουν οριστεί στον PHP κώδικα και συνιστούν τα σημεία από τα οποία θα ξεκινήσει και θα ολοκληρωθεί η αναζήτηση της συντομότερης διαδρομής.

Το δεύτερο ερώτημα που περιλαμβάνει ο PHP κώδικας συνιστά ένα ερώτημα συντομότερης διαδρομής, το οποίο μεταβιβάζεται στη βάση για την εκκίνηση της διαδικασίας εύρεσης συντομότερης διαδρομής, σύμφωνα με τα δεδομένα εισόδου που κάθε φορά ορίζει ο χρήστης από την ιστοσελίδα και τα οποία αφορούν τον αλγόριθμο δρομολόγησης, το κριτήριο βάσει του οποίου πρόκειται να υπολογιστεί το βάρος της διαδρομής και τη συμμετοχή ή μη στους υπολογισμούς των τμημάτων του οδικού δικτύου που βρίσκονται στην περιοχή του δακτυλίου. Το ερώτημα SQL που αποστέλλεται στη βάση, ενεργοποιεί κάθε φορά την αντίστοιχη συνάρτηση που υλοποιεί τον αλγόριθμο δρομολόγησης που ο χρήστης έχει επιλέξει για την εύρεση συντομότερης διαδρομής.

Τα επόμενα τμήματα του PHP κώδικα αφορούν τη λήψη των αποτελεσμάτων από τη βάση, την εισαγωγή και την αποθήκευσή τους στο XML και το KML έγγραφο που έχουν

δημιουργηθεί, την αποδέσμευση της μνήμης από τα αποτελέσματα του *query* που μόλις εκτελέστηκε με τη βοήθεια της PHP συνάρτησης *pg_free_result()* και το κλείσιμο της σύνδεσης με τη βάση δεδομένων με τη βοήθεια της PHP συνάρτησης *pg_close()*.

Ο κώδικας PHP που αναλύθηκε στις παραπάνω παραγράφους, αφορά την εύρεση συντομότερης διαδρομής στην περίπτωση που ο χρήστης επιλέγει ως αλγόριθμο δρομολόγησης τον αλγόριθμο του Dijkstra ή τον αλγόριθμο A*. Στην περίπτωση επιλογής του αλγόριθμου TSP και λόγω των διαφορετικών ορισμάτων που δέχεται η αντίστοιχη συνάρτηση, δημιουργήθηκε ένας δεύτερος PHP κώδικας με την ίδια λογική που δημιουργήθηκε και ο πρώτος. Οι διαφοροποιήσεις που εμφανίζει σε σχέση με τον πρώτο, εντοπίζονται στο δεύτερο ερώτημα που αφορά τη συνάρτηση δρομολόγησης και τις διαφορετικές παραμέτρους που λαμβάνει ως ορίσματα, καθώς και στην εισαγωγή περισσότερων από δύο σημείων, οι συντεταγμένες των οποίων πρέπει να ληφθούν υπόψη κατά τη διαδικασία της δρομολόγησης. Πιό συγκεκριμένα, ο κώδικας αποθηκεύει τα σημεία που επιλέγει ο χρήστης στο χαρτογραφικό υπόβαθρο και αφού βρει τα πλησιέστερα σε αυτά σημεία, που βρίσκονται αποθηκευμένα στη βάση, τα μετατρέπει σε ένα ενιαίο αλφαριθμητικό το οποίο και εισάγεται στο αντίστοιχο SQL ερώτημα δρομολόγησης. Στη συνέχεια και σύμφωνα με την ίδια ακριβώς διαδικασία που ισχύει για το αλγόριθμους Dijkstra και A*, μεταβιβάζεται στη βάση ένα ερώτημα δρομολόγησης βάσει του οποίου ενεργοποιείται η συνάρτηση *tsp_sp_directed_cost* και υλοποιούνται οι αντίστοιχοι υπολογισμοί. Οι δύο επιμέρους PHP κώδικες που δημιουργήθηκαν για τις περιπτώσεις δρομολόγησης με τους αλγόριθμους Dijkstra, A* και TSP είναι οι κώδικες *routing.php* και *routingtsp.php* αντίστοιχα και βρίσκονται στο Παράρτημα.

ΚΕΦΑΛΑΙΟ 6: ΣΥΜΠΕΡΑΣΜΑΤΑ

Ο στόχος της παρούσας εργασίας ήταν η υλοποίηση μιας διαδικτυακής εφαρμογής δρομολόγησης μέσω της οποίας, ένας χρήστης του οδικού δικτύου της Αθήνας θα έχει τη δυνατότητα εύρεσης της συντομότερης διαδρομής που πρέπει να ακολουθήσει για τη μετάβασή του από ένα σημείο του δικτύου σε κάποιο άλλο. Ο χρήστης, μέσω ενός διαδικτυακού χάρτη, επιλέγει τα σημεία που ορίζουν την προς αναζήτηση διαδρομή, τον επιθυμητό αλγόριθμο δρομολόγησης, το κριτήριο βάσει του οποίου πρόκειται να υπολογιστεί το κόστος διάσχισης της ζητούμενης διαδρομής και την περιοχή κίνησής του εντός ή εκτός του δακτυλίου. Το αίτημά του μεταβιβάζεται στο *server* και μετά την επεξεργασία των δεδομένων στη βάση, λαμβάνει το αποτέλεσμα της αναζήτησης ως λεκτική περιγραφή σε έναν πίνακα και ως γεωμετρική οντότητα σε ένα διαδικτυακό χάρτη.

Για το σκοπό αυτό, κρίθηκε απαραίτητη η δημιουργία μιας χωρικής βάσης δεδομένων σε περιβάλλον PostgreSQL+PostGIS, εμπλουτισμένης με τις συναρτήσεις δρομολόγησης που διαθέτει το pgRouting. Το δίκτυο αναπαρίσταται στη βάση ως γράφος, οι κόμβοι του οποίου συνδέουν τα επιμέρους τμήματα του οδικού δικτύου. Τα τμήματα του οδικού δικτύου συνιστούν τις γεωμετρικές οντότητες που αποθηκεύονται στη στήλη γεωμετρίας του πίνακα όπου καταχωρούνται τα στοιχεία του δικτύου. Συνεπώς, στη βάση δεδομένων αποθηκεύεται το σύνολο των περιγραφικών και γεωμετρικών χαρακτηριστικών των γεωμετρικών οντοτήτων του δικτύου. Επιπλέον, οι ιδιαίτερες ανάγκες της εφαρμογής, οδήγησαν στη δημιουργία νέων συναρτήσεων στο περιβάλλον της PostgreSQL, οι οποίες χρησιμοποιούνται για την εύρεση των αποτελεσμάτων που κάθε φορά αναζητά ο χρήστης της εφαρμογής.

Το PostGIS συνιστά μια επέκταση της PostgreSQL και είναι ειδικά σχεδιασμένο για τη διαχείριση χωρικών δεδομένων καθώς υποστηρίζει την υλοποίηση χωρικών λειτουργιών, με τη βοήθεια των γεωμετρικών τελεστών και των συναρτήσεων που διαθέτει, αξιοποιώντας τον PostgreSQL backend server. Υπακούει στις προδιαγραφές του προτύπου του OGC, παρέχει στο χρήστη τη δυνατότητα ορισμού συστήματος αναφοράς στα δεδομένα του και επιτρέπει την πρόσβαση από χαρτογραφικές εφαρμογές μέσα από τη δυνατότητα προσπέλασης των χωρικών δεδομένων από Java clients καθώς και την

οπτικοποίηση των χωρικών δεδομένων μέσω Map Server και uDIG. Το pgRouting διαθέτει έτοιμες συναρτήσεις, οι οποίες υλοποιούν βασικούς αλγόριθμους δρομολόγησης και μπορούν να χρησιμοποιηθούν για την εύρεση συντομότερης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων ενός δικτύου.

Η ιστοσελίδα που σχεδιάστηκε συνιστά το περιβάλλον μέσα από το οποίο ο χρήστης αλληλεπιδρά με τον *server*, καθορίζει τα δεδομένα εισόδου στον *server* και λαμβάνει τα αποτελέσματα της εκάστοτε αναζήτησης. Ως χαρτογραφικό υπόβαθρο χρησιμοποιήθηκε ένας διαδικτυακός χάρτης της Google Maps. Ο σχεδιασμός της *client* εφαρμογής υλοποιήθηκε σε κώδικα Javascript, ο οποίος ενσωματώθηκε στον κώδικα HTML που σχεδιάστηκε για τη δόμηση και μορφοποίηση της ιστοσελίδας.

Η επικοινωνία ανάμεσα στον *client* και τον *server* επιτυγχάνεται μέσα από τη δημιουργία ενός κώδικα PHP, ο οποίος μεταβιβάζει τα αιτήματα του πελάτη στον εξυπηρετητή και επιστρέφει τα εξαγόμενα ως λεκτική και ως γεωμετρική περιγραφή στο περιβάλλον του χρήστη, με τη βοήθεια δύο ενδιάμεσων αρχείων XML και KML στα οποία καταχωρείται η πληροφορία που προκύπτει μετά την επεξεργασία των δεδομένων εισόδου στη βάση.

Συνεπώς, οι διαδικασίες υλοποίησης της διαδικτυακής εφαρμογής που αναπτύχθηκε στα πλαίσια της παρούσας εργασίας προϋποθέτουν την ανάπτυξη επιμέρους εφαρμογών, κάθε μια από τις οποίες επιτελεί μια συγκεκριμένη λειτουργία. Το τελικό αποτέλεσμα, προκύπτει ως συνδυασμός των επιμέρους εφαρμογών οι οποίες επικοινωνούν μεταξύ τους σε επίπεδο ανταλλαγής πληροφοριών. Τα διαθέσιμα προγραμματιστικά εργαλεία που χρησιμοποιήθηκαν κατά τη διαδικασία ανάπτυξης της εφαρμογής συνιστούν ελεύθερα λογισμικά, ανοικτού κώδικα, οι δυνατότητες των οποίων ανταποκρίθηκαν πλήρως στις απαιτήσεις της σχεδιαζόμενης εφαρμογής. Ωστόσο, απαιτήθηκε ο σχεδιασμός ορισμένων νέων συναρτήσεων στο pgRouting, ώστε να ανταποκρίνονται πλήρως στις απαιτήσεις του χρήστη επιστρέφοντάς του κάποια επιπρόσθετα στοιχεία σχετικά με τη διαδρομή που πρέπει να ακολουθήσει για τη μετάβασή του από ένα σημείο του δικτύου σε κάποιο άλλο. Οι νέες συναρτήσεις που δημιουργήθηκαν συνιστούν επεκτάσεις των συναρτήσεων που διαθέτει έτοιμες το pgRouting. Τόσο το διαδικτυακό περιβάλλον που χρησιμοποιήθηκε για το σχεδιασμό της ιστοσελίδας, όσο και το περιβάλλον υλοποίησης της χωρικής βάσης δεδομένων είναι φιλικά προς το χρήστη και σχετικά απλά για την ανάπτυξη των αντίστοιχων εφαρμογών.

Συμπερασματικά, τα προγραμματιστικά εργαλεία που διατίθενται για το σχεδιασμό διαδικτυακών εφαρμογών και την ανάπτυξη εφαρμογών βάσεων δεδομένων, επαρκούν για την ανάπτυξη απλών και αποτελεσματικών εφαρμογών δρομολόγησης οι οποίες

ανταποκρίνονται στις απαιτήσεις των χρηστών για πλοήγηση. Ωστόσο, η ραγδαία ανάπτυξη και εξάπλωση των υπηρεσιών πλοήγησης με τη βοήθεια διαδικτυακών χαρτών, απαιτεί τη διαρκή ανανέωση και ενίσχυση των διαθέσιμων προγραμματιστικών εργαλείων με περισσότερες δυνατότητες που ανταποκρίνονται στις διαρκώς αυξανόμενες ανάγκες των χρηστών. Τέλος, εξαιρετικά σημαντικό είναι και το ζήτημα της ελεύθερης διάθεσης χωρικών δεδομένων (αρχεία τύπου shapefile, αρχεία kml κ.λπ.) που απαιτούνται για την ανάπτυξη αυτού του είδους των εφαρμογών, καθώς ένας σημαντικός αριθμός χρηστών δεν έχει πρόσβαση σε υψηλής ακρίβειας και ποιότητας χωρικά δεδομένα.

6.1 Πλεονεκτήματα PostgreSQL/PostGIS + pgRouting

Η PostgreSQL και το PostGIS συνιστούν λογισμικά ανοικτού κώδικα, στα οποία έχει ελεύθερη πρόσβαση οποιοσδήποτε χρήστης του παγκόσμιου ιστού και οι δυνατότητες που παρέχουν είναι εφάμιλλες με τις δυνατότητες που παρέχουν στο χρήστη τα αντίστοιχα εμπορικά συστήματα ανάπτυξης χωρικών βάσεων δεδομένων. Επιπλέον, επιτρέπεται η ενσωμάτωσή τους σε εφαρμογές που αναπτύσσονται από τον εκάστοτε χρήστη, ενώ συνιστούν δύο λογισμικά με πλήρη διαδικτυακό προσανατολισμό καθώς υλοποιούν συνδέσεις client – server μέσω των διεπαφών που διαθέτουν για διάφορες γλώσσες προγραμματισμού (C/C++, Java, Python κ.λπ.). Είναι συμβατά με το σύνολο των λειτουργικών συστημάτων και υποστηρίζουν ένα πλήθος γεωμετρικών συναρτήσεων και τελεστών για την υλοποίηση χωρικών λειτουργιών. Παράλληλα, μπορούν να διαχειριστούν ένα μεγάλο αριθμό ταυτόχρονων χρηστών διασφαλίζοντας την απαιτούμενη αξιοπιστία ως προς την ορθότητα υλοποίησης διάφορων λειτουργιών. Η δομή GiST που χρησιμοποιείται για τη δεικτοδότηση των δεδομένων, παρέχει τη δυνατότητα συνδυασμού διαφορετικών αλγόριθμων ταξινόμησης και αναζήτησης όπως τα B- trees, τα R- trees κ.ά. Το PostGIS υποστηρίζει το σύνολο των χωρικών τύπων και των χωρικών λειτουργιών όπως ορίζονται από το OGC, επεκτείνοντας τις δυνατότητες της PostgreSQL ως προς τη διαχείριση χωρικών δεδομένων. Παρέχει στο χρήστη τη δυνατότητα ορισμού συστήματος αναφοράς στα δεδομένα του, ενώ στον πίνακα *SPATIAL_REF_SYS* διαθέτει υλοποιημένα και κωδικοποιημένα όλα τα γνωστά συστήματα αναφοράς με την πλήρη περιγραφή τους. Τέλος, επιτρέπει την οπτικοποίηση των χωρικών δεδομένων που βρίσκονται αποθηκευμένα στη βάση μέσα από τη δυνατότητα σύνδεσής του με λογισμικά οπτικοποίησης χωρικών δεδομένων.

Το pgRouting διαθέτει στο χρήστη έτοιμες συναρτήσεις, οι οποίες υλοποιούν αλγόριθμους δρομολόγησης και επιλύουν το πρόβλημα εύρεσης συντομότερης διαδρομής. Παρέχει τη δυνατότητα οπτικοποίησης των αποτελεσμάτων που προκύπτουν μέσα από τη σύνδεσή του με περιβάλλοντα γραφικής αναπαράστασης χωρικών δεδομένων, ενώ το κόστος διάσχισης της συντομότερης διαδρομής μπορεί να υπολογιστεί δυναμικά μέσω κώδικα SQL και η τιμή του μπορεί να προκύψει από πολλαπλές εγγραφές ή πίνακες.

6.2 Συγκριτική Αξιολόγηση της Εφαρμογής με Λογισμικά GIS

Η ραγδαία ανάπτυξη των τεχνολογιών διαχείρισης χωρικής πληροφορίας οδήγησε στο σχεδιασμό αντίστοιχων πακέτων λογισμικού, που παρέχουν στους χρήστες δυνατότητες διαχείρισης πάσης φύσεως χωρικών δεδομένων. Τα λογισμικά που διατίθενται για το σκοπό αυτό διαφοροποιούνται, τόσο ως προς την αρχιτεκτονική τους όσο και ως προς τις δυνατότητες που παρέχουν για την επεξεργασία των χωρικών δεδομένων.

Στα πλαίσια της παρούσας εφαρμογής, οι διαδικασίες διαχείρισης και επεξεργασίας των χωρικών δεδομένων λαμβάνουν χώρα σε μια βάση δεδομένων για το σχεδιασμό της οποίας χρησιμοποιήθηκε το ΣΔΧΒΔ PostgreSQL/ PostGIS εμπλουτισμένο με τις συναρτήσεις δρομολόγησης του pgRouting. Το συγκεκριμένο ΣΔΧΒΔ συνιστά λογισμικό ανοικτού κώδικα, το οποίο διατίθεται δωρεάν και παρέχει στο χρήστη εφάμιλλες δυνατότητες με αυτές που προσφέρουν αντίστοιχα εμπορικά συστήματα. Παράλληλα, υπάρχει η δυνατότητα επεξεργασίας των ήδη υλοποιημένων συναρτήσεων που το σύστημα αυτό διαθέτει και η δημιουργία νέων, γεγονός που κατέστησε δυνατή την υλοποίηση της παρούσας εφαρμογής. Συνεπώς, ο χρήστης μπορεί να παρέμβει στον κώδικα και να δημιουργήσει συναρτήσεις οι οποίες ανταποκρίνονται στις ιδιαίτερες ανάγκες της εκάστοτε εφαρμογής. Θα πρέπει να σημειωθεί ότι η απουσία της συγκεκριμένης δυνατότητας θα καθιστούσε αδύνατη την ολοκλήρωση της εφαρμογής που σχεδιάστηκε και ως εκ τούτου θα έπρεπε να αναζητηθούν διαφορετικές λύσεις. Παράλληλα, αξιοποιήθηκε η δυνατότητα σύνδεσης του συστήματος PostgreSQL/ PostGIS με λογισμικά οπτικοποίησης προκειμένου να ελεγχθεί η ορθότητα των εξαγόμενων αποτελεσμάτων, αλλά και να αναπαρασταθούν τα αποτελέσματα ως γεωγραφική οντότητα επάνω στο χάρτη της ιστοσελίδας.

Σε αντίθεση με την Oracle Spatial, το πακέτο PostgreSQL/PostGIS δε διαθέτει ένα αυστηρά υλοποιημένο μοντέλο δεδομένων δικτύου για τη μοντελοποίηση και ανάλυση χωρικών δικτύων. Συνεπώς, δεν υπάρχει δυνατότητα διεξαγωγής ορισμένων απευθείας ελέγχων που σχετίζονται με τη φύση και τα χαρακτηριστικά του εκάστοτε δικτύου, όπως για παράδειγμα ο έλεγχος συνδεσιμότητας μεταξύ των κόμβων του δικτύου. Ωστόσο, η Oracle απαιτεί για την εγκατάσταση και τη λειτουργία της, τη διάθεση πολύ περισσότερου χώρου μνήμης σε σχέση με την PostgreSQL/PostGIS. Η αρχιτεκτονική της Oracle είναι πιο σύνθετη από αυτήν του PostGIS καθώς είναι εμπλουτισμένη με περισσότερα εργαλεία. Ωστόσο, η διαδικασία διεξαγωγής ερωτημάτων στο PostGIS είναι ταχύτερη. Το μοντέλο που χρησιμοποιεί η Oracle Spatial για την αναπράσταση ενός χωρικού δικτύου αναπαριστά το δίκτυο, είτε ως μια σχεσιο- αντικειμενοστρεφή οντότητα στη βάση, είτε ως ένα αντικείμενο Java στο επίπεδο του *client* ή της εφαρμογής. Παράλληλα, διαθέτει μια σειρά από σύνθετες χωρικές συναρτήσεις και πολύπλοκους χωρικούς τελεστές που αξιοποιούνται ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Το σύστημα PostgreSQL/PostGIS δεν αναπαριστά με τον ίδιο τρόπο τα χωρικά δεδομένα, ούτε διαθέτει το πλήθος των συναρτήσεων της Oracle Spatial. Ωστόσο, διαθέτει ένα μεγάλο αριθμό χωρικών τελεστών και συναρτήσεων για την υλοποίηση διαφόρων λειτουργιών [URL22].

Το ArcGIS Network Analyst συνιστά ένα εναλλακτικό εργαλείο που παρέχει δυνατότητες χωρικής ανάλυσης, όπως ο υπολογισμός διαδρομών μεταξύ ενός σημείου αφετηρίας και ενός σημείου προορισμού, η εύρεση διαδρομής που διέρχεται από περισσότερα των δύο σημείων, η ανάθεση κατεύθυνσης στις ακμές ενός δικτύου, η αναζήτηση εγγύτερων σημείων επάνω σε ένα χάρτη, η αναπαράσταση περιοχών ενδιαφέροντος με τη βοήθεια πολυγώνων κ.λπ. Το εργαλείο αυτό συνιστά μια επέκταση του ArcGIS, για τη διεξαγωγή αναλυτικών λειτουργιών σε χωρικά δίκτυα καθώς το ArcGIS παρέχει στο χρήστη τη δυνατότητα υλοποίησης μιας σειράς διαδικασιών σε έναν περιορισμένο αριθμό μεμονωμένων αντικειμένων (objects) ή θεματικών επιπέδων (layers) [URL23].

Στα εργαλεία που χρησιμοποιούνται για την ανάλυση χωρικών δικτύων ανήκει επίσης το Google Directions Map API, μια υπηρεσία που χρησιμοποιείται για τον υπολογισμό κατευθύνσεων μεταξύ σημείων στο χώρο μέσα από τη διατύπωση ενός *http* αιτήματος. Το Google Directions API δεν ανταποκρίνεται σε αιτήματα χρηστών σε πραγματικό χρόνο, τα οποία μάλιστα συνοδεύονται από ένα συγκεκριμένο input, όπως το Javascript API που χρησιμοποιήθηκε στα πλαίσια της παρούσας εφαρμογής [URL3]. Άλλα εργαλεία χωρικής ανάλυσης είναι ο Geomedia Transportation Manager της Intergraph, το

eRouteLogistics, Συστήματα Χωρικών Βάσεων Δεδομένων με χωρικές επεκτάσεις όπως η MySQL Spatial κ.ά.

Συμπερασματικά, τα εργαλεία που διατίθενται για τη διαχείριση χωρικών δεδομένων διαφοροποιούνται ως προς τις δυνατότητες και τα χαρακτηριστικά τους. Η επιλογή κάποιου από αυτά εξαρτάται κάθε φορά από το βαθμό στον οποίο ανταποκρίνεται στις απαιτήσεις της εκάστοτε εφαρμογής και από το επιδιωκώμενο αποτέλεσμα. Για το λόγο αυτό, μια συγκριτική θεώρηση μεταξύ τους δεν έχει νόημα στο επίπεδο της αξιολόγησης για την ανάδειξη του βέλτιστου, αλλά στο σχηματισμό μιας πλήρους εικόνας των δυνατοτήτων καθενός από αυτά, ούτως ώστε να επιλεγεί το πλέον κατάλληλο σε συνάρτηση πάντα με τις απαιτήσεις της εφαρμογής που πρόκειται να υλοποιηθεί.

6.3 Μελλοντικές Προεκτάσεις

Η εφαρμογή που σχεδιάστηκε, παρέχει στο χρήστη τη δυνατότητα αξιοποίησης τριών διαφορετικών αλγόριθμων δρομολόγησης για την εύρεση της βέλτιστης διαδρομής μεταξύ δύο ή περισσοτέρων σημείων του δικτύου, στη βάση ορισμένων κριτηρίων. Τα αποτελέσματα που εξάγονται είναι αξιόπιστα και σε ικανοποιητικό βαθμό ακριβή, ενώ η περιγραφή της εκάστοτε διαδρομής που προκύπτει είναι λεπτομερής και αρκετά κατατοπιστική. Ωστόσο, μερικές πρόσθετες προσθήκες είναι δυνατό να συνεισφέρουν στη βελτίωση της εφαρμογής, ούτως ώστε να γίνει πιο ελκυστική και να εμπλουτιστεί ως προς τις δυνατότητες που ήδη παρέχει στο χρήστη.

Η προσθήκη πιο εξειδικευμένων αλγόριθμων διάσχισης γράφων, εμπλουτισμένων με αποδοτικότερες τεχνικές δρομολόγησης, θα είχε ως αποτέλεσμα την εξαγωγή διαφοροποιημένων και ακριβέστερων αποτελεσμάτων. Παράλληλα, ο εμπλουτισμός των χωρικών δεδομένων που βρίσκονται στη βάση, με δεδομένα που αναπαριστούν ένα ευρύτερο οδικό δίκτυο από αυτό της Αθήνας, θα παρείχε στο χρήστη τη δυνατότητα αναζήτησης συντομότερης διαδρομής σε μια μεγαλύτερη χωρική έκταση. Επιπρόσθετα, η παράμετρος βάσει της οποίας υπολογίζεται το κόστος διάσχισης της εκάστοτε διαδρομής θα μπορούσε να λάβει και άλλες τιμές όπως για παράδειγμα το κόστος διοδίων, το κόστος εισιτηρίου εάν πρόκειται για μετακίνηση με δημόσιες συγκοινωνίες, την ελάχιστη κατανάλωση καυσίμων κ.α. Τέλος, θα είχε αρκετό ενδιαφέρον η δυνατότητα από πλευράς του χρήστη να καθορίζει τον τρόπο με τον οποίο επιθυμεί να μετακινηθεί μέσα στο οδικό δίκτυο, εάν δηλαδή επιθυμεί να μετακινηθεί με το ιδιωτικό του όχημα ή με τα Μέσα Μαζικής Μεταφοράς καθώς και η αξιοποίηση στοιχείων κυκλοφορίας όπως οι ταχύτητες

οχημάτων από στίγματα GPS, η χρονική καθυστέρηση που οφείλεται στους φωτεινούς σηματοδότες που ρυθμίζουν την κυκλοφορία κ.ά.

Σε κάθε περίπτωση, μια εφαρμογή που σχεδιάζεται για την εξυπηρέτηση του κοινού θα πρέπει να είναι σχετικά απλή και φιλική ως προς τη χρήση της, να δίνει ακριβή και σαφή αποτελέσματα στα οποία μπορεί να βασιστεί ο χρήστης και να εξασφαλίζει τη γρήγορη και αξιόπιστη εξαγωγή των εκάστοτε αποτελεσμάτων.

Βιβλιογραφία

Βιβλιογραφικές Αναφορές

[1] Aldous M. Joan, Wilson J. Robin (2000), **Graphs and Applications: An Introductory Approach**, Springer Publications, London.

[2] Bechnke Kai, Turkow Florian (2007), *pgRouting and the UMN MapServer*, Creative Commons, Deutschland.

[3] Cormen H. Thomas, Leiserson E. Charles, Rivest L. Ronald and Stein Clifford (2001), **Introduction to Algorithms**, The MIT Press, Massachussets Institute of Technology.

[4] Egenhofer J. Max (1994), *Spatial SQL: A Query and Presentation Language*, IEEE Transactions on Knowledge and Data Engineering, 6 (1): 86-95.

[5] Gaikwad Shantaram Santosh (2008), *Using PHP/ PostgreSQL/ PostGIS with Google Maps*, Open Source Geospatial Tools Workshop, 28th-30th August 2008, OSGeo, India.

[6] Golden Bruce (1975), *Shortest Path Algorithms: A Comparison*, Operations Research, Vol.24, No.6, pp.1164-1168, M.I.T- Boston.

[7] Goodchild Michael, Haining Robert, Wise Stephen and 12 others (1992), *Integrating GIS and Spatial Data Analysis: Problems and Possibilities*, Int. J. Geographical Information Systems, vol. 6, No. 5, pp. 407-423.

[8] Guting Hartmut Ralf (1994), *An Introduction to Spatial Database Systems*, Invited Contribution to a Special Issue on Spatial Database Systems of the VLDB Journal, Vol. 3, No. 4.

[9] Guting Hartmut Ralf (1994), Spatial Database Systems: Tutorial Notes, Fern Universitat Hagen, Praktische Informatik IV, D-58084, Germany.

[10] Hjaltason R. Gisli, Samet Hanan (1998), *Ranking in Spatial Databases*, Computer Science Department and Center for Automation Research and Institute for Advanced Computer Studies, University of Maryland, Maryland- U.S.A.

[11] Kainz Wolfgang, Egenhofer J. Max and Greasley Ian (1993), *Modeling Spatial Relations and Operations with Partially Ordered Sets*, International Journal of Geographical Information Systems, 7(3): 215-229.

[12] Kastl Daniel, Junod Frederic (2010), Routing with pgRpouting Tools, OpenStreetMap Road Data and GeoExt Manual (Release 1), Workshop- FOSS4G.

[13] Langeman Matt (2005), *The Network of Weblogs and a Review of Real and Social Network Research*, Msci 620, University of Waterloo.

[14] Paredaens Jan, Kuijpers Bart (1998), Data Models and Query Languages for Spatial Databases, University of Antwerp.

[15] PostgreSQL User Guide, Release 8.3, The PostgreSQL Global Development Group.

[16] Quantum GIS User Guide, Version 1.5.0. 'Tethys'.

[17] Samet Hanan (1995), *Spatial Data Structures in Modern Database Systems: The Object Model, Interoperability and Beyond*, (W. Kim Ed.), Addison- Wesley/ACM Press, 1995, pp. 361-385.

[18] Samet Hanan, Aref G. Walid (1994), *Spatial Data Models and Query Processing*, Modern Database Systems, Book Chapter, (W. Kim, Ed.), Addison-Wesley/ACM Press, pp.338-360.

[19] Schneider Markus, Behr Thomas (2006), *Topological Relationships between Complex Spatial Objects*, ACM Transactions on Database Systems, Vol. 31, No. 1, pp. 39-81.

[20] Shekhar Shashi, Chawla Sanjay (2003), **Spatial Databases: A Tour**, Pearson Education Inc., Upper Saddle River, New Jersey 07458.

[21] Shekhar S., Chawla S., Ravada S., Fetterer A., Liu X. and Lu C.T. (1999), *Spatial Databases: Accomplishments and Research Needs*, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No 1, pp. 45-55.

[22] Van Oosterom Peter, Stoter Jantien, Quak Wilko and Zlatanova Sisi (2002), *The Balance between Geometry and Topology*, Proc. of the 10th Int. Symposium on Spatial Data Handling, pp.209-224, SDH'02.

[23] Vaswani Vikram (2000 – 2002), *PHP and PostgreSQL*, Free PDF Manual Guide.

[24] Zhan F. Benjamin, Noon E. Charles (1998), *Shortest Path Algorithms: An Evaluation using Real Road Networks*, Transportation Science, Vol. 32, pp.65-73.

[25] Κολιός Ν. (2009), *Χωρική Βάση Δεδομένων PostgreSQL/ PostGIS και Σύστημα Γεωγραφικών Πληροφοριών Quantum GIS*, Οδηγός Χρήσης, Εταιρεία Ελεύθερου Λογισμικού/ Λογισμικού Ανοικτού Κώδικα- Creative Commons Attribution- Non Commercial- ShareAlike 3.0- Ελλάδα

[26] Πατρούμπας Κ. (2008), Διδακτικές Σημειώσεις Μαθήματος: Χωρικές Βάσεις Δεδομένων, ΠΜΣ Γεωπληροφορικής, Σ.Α.Τ.Μ. – Ε.Μ.Π., Αθήνα.

[27] Σελλής Τ. (2006), Διδακτικές Σημειώσεις Μαθήματος: Χωρικές Βάσεις Δεδομένων, ΠΜΣ Γεωπληροφορικής, Σ.Α.Τ.Μ. – Ε.Μ.Π., Αθήνα.

[28] Στεφανάκης Εμμανουήλ (2003), **Βάσεις Γεωγραφικών Δεδομένων και Συστήματα Γεωγραφικών Πληροφοριών**, Εκδόσεις Παπασωτηρίου, Αθήνα.

Δικτυακοί Τόποι

[URL1] Algorithms and Data Structures:
http://www.algolist.net/Data_structures/Graph

[URL2] Beng Chin Ooi, Ron Sacks- Davis, Jiawei Han, Indexing in Spatial Databases:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.9499&rep

[URL3] Google Maps API Official Site:
http://code.google.com/apis/maps/index.html

[URL4] Kent State University, Traveling Salesman Problem Heuristic:
http://www.personal.kent.edu/~rmuhamma/Compgeometry/MyCG/CG-Applets/TSP/notspcli.htm

[URL5] LexiconCS:
http://lexiconcs.wikispaces.com

[URL6] MapServer Official Site:
http://mapserver.org/

[URL7] McSlick's Chronic Paper Jam:
http://mcslick.wordpress.com/2009/08/24/the-seven-bridges-of-konigsberg/

[URL8] OSGeo Live:
http://live.osgeo.org/en/overview/pgrouting_overview.html

[URL9] pgRouting Official Site:
http://www.pgrouting.org/

[URL10] PHP Manual:
http://php.net/manual/en/tutorial.php

[URL11] PHP Tutorial:
http://www.tizag.com/phpT/

[URL12] PostGIS Manual 1.3.6:
http://postgis.refractions.net/documentation/manual-1.3/

[URL13] PostGIS Official Site:
http://postgis.refractions.net/

[URL14] PostgreSQL Official Site:
http://www.postgresql.org/

[URL15] University of Cambridge - Department of Engineering:
http://www-sigproc.eng.cam.ac.uk/~atc27/matlab/layout.html.

[URL16] Wikipedia:
http://en.wikipedia.org/wiki/Graph_(mathematics),
http://en.wikipedia.org/wiki/Travelling_salesman_problem

[URL17] Wolfram MathWorld- The web's most extensive mathematics resource:
http://mathworld.wolfram.com/ConnectedGraph.html,
http://mathworld.wolfram.com/CycleGraph.htm

[URL18] w3schools- the world's largest web development site:
http://www.w3schools.com/

[URL19] Ιστοσελίδα Μαθήματος: Αλγόριθμοι και Πολυπλοκότητα, Ακαδημαϊκό Έτος 2010-2011, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ Ε.Μ.Π.: http://www.corelab.ntua.gr/courses/algorithms/slides/NewGraphs2.pdf

[URL20] Ιστοσελίδα Μαθήματος: Προγραμματιστικές Τεχνικές, Ακαδημαϊκό Έτος 2007-2008, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ Ε.Μ.Π.: http://courses.softlab.ntua.gr/progtech/2004a/graphs.pdf.

[URL21] Ιστοσελίδα Πανεπιστημίου "DePaul":
http://condor.depaul.edu/ntomuro/courses/416/notes/lecture7-graphshort.html

[URL22] Ιστοσελίδα της Oracle- Official Oracle Site
http://www.oracle.com/

[URL23] Ιστοσελίδα του ArcGIS Network Analyst- ESRI: ArcGIS Network Analyst
http://www.esri.com/software/arcgis/extensions/networkanalyst/index.html

Γλωσσάριο

Abstract Data Type (ADT)	Αφηρημένος Τύπος Δεδομένων
Adjacency List	Λίστα Γειτνίασης
Adjacent Edges	Γειτονικές Ακμές
Adjacent Nodes	Γειτονικοί Κόμβοι
Application Programming Interface	Διεπαφή
Boolean Value	Λογική Τιμή
Breadth- First Search (BFS)	Κατά Πλάτος Αναζήτηση Γράφου
Client-Server Application	Εφαρμογή Πελάτη- Εξυπηρετητή
Depth- First Search (DFS)	Κατά Βάθος Αναζήτηση Γράφου
Directed Edges	Κατευθυνόμενες Ακμές
Directed Graph	Κατευθυνόμενος Γράφος
Direct Predecessor	Άμεσος Προκάτοχος
Direct Successor	Άμεσος Διάδοχος
Edge	Ακμή
Geometry	Γεωμετρία
GIS	Geographical Information System
Graph	Γράφος
Graph Traversal	Διάσχιση Γράφου
Heuristics	Ευρετικοί Κανόνες
Input Data	Δεδομένα Εισόδου
Multilinestring	Πολυγραμμή
Node	Κόμβος
OGC	Open Geospatial Consortium
Output Data	Δεδομένα Εξόδου
Path	Μονοπάτι
Route	Διαδρομή
Server	Εξυπηρετητής
Shortest Path Problem	Πρόβλημα της Συντομότερης Διαδρομής
Source Node	Αρχικός Κόμβος/ Κόμβος Αφετηρίας
SQL	Structured Query Language

Spatial Analysis	Χωρική Ανάλυση
Spatial Operations	Χωρικές Λειτουργίες
Spatial Query	Χωρικό Ερώτημα
String	Αλφαριθμητικό
Transitive Closure	Μεταβατική Κλειστότητα
Traveling Salesman Problem	Πρόβλημα του Πλανόδιου Πωλητή
Vertex	Κόμβος
ΣΒΔ	Σύστημα Βάσεων Δεδομένων
ΣΓΠ	Σύστημα Γεωγραφικών Πληροφοριών
ΣΔΧΒΔ	Σύστημα Διαχείρισης ΧΒΔ
ΧΒΔ	Χωρική Βάση Δεδομένων

Παράρτημα

I. Κώδικας δημιουργίας νέου τύπου δεδομένων *'geomsc'*

Στα πλαίσια δημιουργίας νέων συναρτήσεων στην PostgreSQL, κρίθηκε απαραίτητη η δημιουργία ενός νέου τύπου δεδομένων, του τύπου *'geomsc'*. Συνεπώς, τα αποτελέσματα κάθε μιας από τις συναρτήσεις που παρουσιάζονται στο εδάφιο ΙΙ του παρόντος παραρτήματος είναι τύπου *'geomsc'*. Η δημιουργία του νέου τύπου δεδομένων έγινε με τη βοήθεια κώδικα PL/pgSQL.

```
DROP TYPE geomsc;

CREATE TYPE geomsc AS (
        id integer,
        gid integer,
        cost double precision,
        greekname varchar(50),
        latinname varchar(50),
        the_geom geometry
);


ALTER TYPE public.geomsc OWNER TO postgres;
```

II. Κώδικες δημιουργίας νέων συναρτήσεων σε περιβάλλον PostgeSQL/ PostGIS + pgRouting

Οι ιδιαίτερες ανάγκες της σχεδιαζόμενης εφαρμογής επέβαλαν τη δημιουργία τεσσάρων νέων συναρτήσεων στη βάση δεδομένων. Οι νέες συναρτήσεις επεκτείνουν τις δυνατότητες των ήδη υφιστάμενων συναρτήσεων και ο προγραμματισμός τους υλοποιήθηκε σε PL/pgSQL.

Κώδικας συνάρτησης 'dijkstra_sp_directed_cost'

```
-- Function: dijkstra_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying)

-- DROP FUNCTION dijkstra_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying);

CREATE OR REPLACE FUNCTION
dijkstra_sp_directed_cost(geom_table character varying, source
integer, target integer, dir boolean, rc boolean, cost_field
character varying, seq character varying)
  RETURNS SETOF geomsc AS
$BODY$
DECLARE
        r record;
        path_result record;
        v_id integer;
        e_id integer;
        geom geomsc;
        query text;
        cost_attr text;
        id integer;

BEGIN

        id :=0;

        IF cost_field = 'cost' THEN
           cost_attr := 'cost';
        ELSE
           cost_attr := 'length::double precision as cost';
        END IF;

        query := 'SELECT oid, gid, R.cost, greekname,
latinname, the_geom FROM (' ||
          'SELECT nextval(' ||  '' || quote_literal(seq) || ''
|| ') AS oid, *  FROM shortest_path(''SELECT gid AS id,
source::integer, target::integer, ' ||
```

```
          cost_attr;

        IF rc THEN query := query || ', reverse_cost ';
        END IF;

        query := query || 'FROM ' ||  quote_ident(geom_table)
|| ''', ' || quote_literal(source) ||
          ' , ' || quote_literal(target) || ' ,
'''||text(dir)||''', '''||text(rc)||''')) R, ' ||
          quote_ident(geom_table) || ' where edge_id = gid
order by oid';

        FOR path_result IN EXECUTE query
        LOOP

                geom.gid      := path_result.gid;
                geom.the_geom := path_result.the_geom;
                geom.cost     := path_result.cost;
                geom.greekname:= path_result.greekname;
                geom.latinname:= path_result.latinname;
            id := id+1;
            geom.id       := id;

                RETURN NEXT geom;

        END LOOP;
        RETURN;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE STRICT
  COST 100
  ROWS 1000;
ALTER FUNCTION dijkstra_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying) OWNER TO postgres;
```

Κώδικας συνάρτησης 'astar_sp_directed_cost'

```
-- Function: astar_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying)

-- DROP FUNCTION astar_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying);

CREATE OR REPLACE FUNCTION astar_sp_directed_cost(geom_table
character varying, source integer, target integer, dir
boolean, rc boolean, cost_field character varying, seq
character varying)
  RETURNS SETOF geomsc AS
```

```
$BODY$
DECLARE
        r record;
        path_result record;
        v_id integer;
        e_id integer;
        geom geomsc;

        query text;
        cost_attr text;

        id integer;
BEGIN

        id :=0;

        IF cost_field = 'cost' THEN
           cost_attr := 'cost';
        ELSE
           cost_attr := 'length::double precision as cost';
        END IF;

        query := 'SELECT oid, gid, R.cost, greekname,
latinname, the_geom FROM (' ||
           'SELECT nextval(' ||  '' || quote_literal(seq) ||
'' || ') AS oid, * FROM shortest_path_astar(''SELECT gid as
id, source::integer, ' ||
           'target::integer, ' || cost_attr   || ', ' ||
           'x1::double precision, y1::double precision, ' ||
           'x2::double precision, y2::double precision ';

        IF rc THEN query := query || ' , reverse_cost ';
        END IF;

        query := query || 'FROM ' || quote_ident(geom_table) ||
' '', ' ||
           quote_literal(source) || ' , ' ||
           quote_literal(target) || ' , ''''||text(dir)||'''',
''''||text(rc)||''''')) R, ' ||
           quote_ident(geom_table) || ' where edge_id = gid
order by oid';

        FOR path_result IN EXECUTE query
        LOOP

              geom.gid      := path_result.gid;
              geom.the_geom := path_result.the_geom;
              geom.cost     := path_result.cost;
              geom.greekname:= path_result.greekname;
              geom.latinname:= path_result.latinname;
              id := id+1;
              geom.id       := id;

              RETURN NEXT geom;
```

```
        END LOOP;
        RETURN;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE STRICT
  COST 100
  ROWS 1000;
ALTER FUNCTION astar_sp_directed_cost(character varying,
integer, integer, boolean, boolean, character varying,
character varying) OWNER TO postgres;
```

Κώδικας συνάρτησης 'dijkstra_sp_delta_directed_cost'

```
-- Function: dijkstra_sp_delta_directed_cost(character
varying, integer, integer, double precision, boolean, boolean,
character varying, character varying)

-- DROP FUNCTION dijkstra_sp_delta_directed_cost(character
varying, integer, integer, double precision, boolean, boolean,
character varying, character varying);

CREATE OR REPLACE FUNCTION
dijkstra_sp_delta_directed_cost(character varying, integer,
integer, double precision, boolean, boolean, character
varying, character varying)
  RETURNS SETOF geomsc AS
$BODY$
DECLARE

        geom_table ALIAS FOR $1;
        sourceid ALIAS FOR $2;
        targetid ALIAS FOR $3;
        delta ALIAS FOR $4;
        dir ALIAS FOR $5;
        rc ALIAS FOR $6;
        cost_field ALIAS FOR $7;
        seq ALIAS FOR $8;


        rec record;
        r record;
        path_result record;
        v_id integer;
        e_id integer;
        geom geomsc;
        cost_attr text;

        srid integer;

        source_x float8;
        source_y float8;
```

```
        target_x float8;
        target_y float8;

        ll_x float8;
        ll_y float8;
        ur_x float8;
        ur_y float8;

        query text;
        id integer;
BEGIN

        id :=0;

        IF cost_field = 'cost' THEN
           cost_attr := 'cost';
        ELSE
           cost_attr := 'length::double precision as cost';
        END IF;

        FOR rec IN EXECUTE
            'select srid(the_geom) from ' ||
            quote_ident(geom_table) || ' limit 1'
        LOOP
        END LOOP;
        srid := rec.srid;

        FOR rec IN EXECUTE
            'select x(startpoint(the_geom)) as source_x from '
||
            quote_ident(geom_table) || ' where source = ' ||
            sourceid ||  ' or target='||sourceid||' limit 1'
        LOOP
        END LOOP;
        source_x := rec.source_x;

        FOR rec IN EXECUTE
            'select y(startpoint(the_geom)) as source_y from '
||
            quote_ident(geom_table) || ' where source = ' ||
            sourceid ||  ' or target='||sourceid||' limit 1'
        LOOP
        END LOOP;

        source_y := rec.source_y;

        FOR rec IN EXECUTE
            'select x(startpoint(the_geom)) as target_x from '
||
            quote_ident(geom_table) || ' where source = ' ||
            targetid ||  ' or target='||targetid||' limit 1'
        LOOP
        END LOOP;

        target_x := rec.target_x;
```

```
        FOR rec IN EXECUTE
            'select y(startpoint(the_geom)) as target_y from '
||
            quote_ident(geom_table) || ' where source = ' ||
            targetid ||  ' or target='||targetid||' limit 1'
        LOOP
        END LOOP;
        target_y := rec.target_y;


        FOR rec IN EXECUTE 'SELECT CASE WHEN
'||source_x||'<'||target_x||
           ' THEN '||source_x||' ELSE '||target_x||
           ' END as ll_x, CASE WHEN
'||source_x||'>'||target_x||
           ' THEN '||source_x||' ELSE '||target_x||' END as
ur_x'
        LOOP
        END LOOP;

        ll_x := rec.ll_x;
        ur_x := rec.ur_x;

        FOR rec IN EXECUTE 'SELECT CASE WHEN '||source_y||'<'||
            target_y||' THEN '||source_y||' ELSE '||
            target_y||' END as ll_y, CASE WHEN '||
            source_y||'>'||target_y||' THEN '||
            source_y||' ELSE '||target_y||' END as ur_y'
        LOOP
        END LOOP;

        ll_y := rec.ll_y;
        ur_y := rec.ur_y;

        query := 'SELECT oid, gid, R.cost, greekname,
latinname, the_geom FROM (' ||
          'SELECT nextval(' ||  '' || quote_literal(seq) || ''
|| ') AS oid, *  FROM shortest_path(''SELECT gid as id,
source::integer, target::integer, ' ||
          cost_attr;

        IF rc THEN query := query || ' , reverse_cost ';
        END IF;

        query := query || ' FROM ' || quote_ident(geom_table)
|| ' where setSRID(''''BOX3D('||
          ll_x-delta||' '||ll_y-delta||','||ur_x+delta||' '||
          ur_y+delta||')''''::BOX3D, ' || srid || ') &&
the_geom'', ' ||
          quote_literal(sourceid) || ' , ' ||
          quote_literal(targetid) || ' , '''||text(dir)||''',
'''||text(rc)||''' )) R, ' ||
          quote_ident(geom_table) || ' where edge_id = gid
order by oid';
```

```
        FOR path_result IN EXECUTE query
        LOOP
                geom.gid      := path_result.gid;
                geom.the_geom := path_result.the_geom;
                geom.cost := path_result.cost;
                geom.greekname := path_result.greekname;
                geom.latinname := path_result.latinname;

            id := id+1;
            geom.id := id;

                RETURN NEXT geom;

        END LOOP;
        RETURN;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE STRICT
  COST 100
  ROWS 1000;
ALTER FUNCTION dijkstra_sp_delta_directed_cost(character
varying, integer, integer, double precision, boolean, boolean,
character varying, character varying) OWNER TO postgres;
```

Κώδικας συνάρτησης 'tsp_sp_directed_cost'

```
-- Function: tsp_sp_directed_cost(character varying, character
varying, integer, double precision, boolean, boolean,
character varying, character varying)

-- DROP FUNCTION tsp_sp_directed_cost(character varying,
character varying, integer, double precision, boolean,
boolean, character varying, character varying);

CREATE OR REPLACE FUNCTION tsp_sp_directed_cost(geom_table
character varying, ids character varying, source integer,
delta double precision, dir boolean, rc boolean, cost_field
character varying, seq character varying)
  RETURNS SETOF geomsc AS
$BODY$
DECLARE
        r record;
        path_result record;
        v_id integer;
        prev integer;
        geom geomsc;
        cost_attr text;

        query text;
```

```
        id integer;
BEGIN

        id :=0;
        prev := source;

        IF cost_field = 'cost' THEN
                    cost_attr := 'cost';
               ELSE
                    cost_attr := 'length::double precision as
cost';
               END IF;

        query := 'SELECT vertex_id FROM tsp(''select distinct
source::integer as source_id, '||
              'x(startpoint(the_geom)),
y(startpoint(the_geom))';

        IF rc THEN query := query || ' , reverse_cost ';
        END IF;

        query := query || ' from ' || quote_ident(geom_table)
|| ' where source in (' ||
           ids || ')'', '''|| ids  ||''', '|| source  ||')';

        FOR path_result IN EXECUTE query
        LOOP
                v_id = path_result.vertex_id;

                FOR r IN EXECUTE 'SELECT * FROM
dijkstra_sp_delta_directed_cost( ''' ||
                  quote_ident(geom_table)  ||''', '|| v_id
||', '||
                  prev ||','||delta||', '''||text(dir)||''',
'''||text(rc) || ''','' || cost_attr  || ''','''  ||  seq  ||
''') R ' LOOP
                    geom.gid := r.gid;
                    geom.the_geom := r.the_geom;
                    geom.cost := r.cost;
                    geom.greekname := r.greekname;
                    geom.latinname := r.latinname;
                    id := id+1;
                    geom.id       := id;
                    RETURN NEXT geom;
                END LOOP;

        prev = v_id;
        END LOOP;
        RETURN;
END;
$BODY$
  LANGUAGE 'plpgsql' VOLATILE STRICT
  COST 100
  ROWS 1000;
```

```
ALTER FUNCTION tsp_sp_directed_cost(character varying,
character varying, integer, double precision, boolean,
boolean, character varying, character varying) OWNER TO
postgres;
```

III. Ο κώδικας της διαδικτυακής εφαρμογής

Στην ενότητα αυτή παρουσιάζεται ο κώδικας της εφαρμογής. Ο προγραμματισμός της εφαρμογής υλοποιήθηκε με τη βοήθεια κώδικα Javascript, ο οποίος ενσωματώθηκε στον HTML κώδικα της ιστοσελίδας. Τα στοιχεία της ιστοσελίδας εμφανίζονται ως περιεχόμενα ενός πίνακα και οι HTML εντολές μέσω των οποίων καθορίζεται η εμφάνισή τους, βρίσκονται εμφωλευμένες μέσα στις ετικέτες *<TABLE></TABLE>*. Οι ετικέτες *<TITLE></TITLE>* περιλαμβάνουν τον τίτλο της ιστοσελίδας ο οποίος εμφανίζεται στη λίστα αγαπημένων ενός φυλλομετρητή και χρησιμοποιείται από τις μηχανές αναζήτησης προκειμένου να δημιουργήσουν τα ευρετήριά τους. Οι ετικέτες που περιλαμβάνουν τον τίτλο της ιστοσελίδας, βρίσκονται εντός των ετικετών *<HEAD></HEAD>*, οι οποίες περιλαμβάνουν επίσης διάφορες πληροφορίες όπως υπερσυνδέσεις ή βιβλιοθήκες κώδικα που πρέπει να φορτωθούν πριν την ιστοσελίδα. Το κυρίως μέρος του HTML κώδικα βρίσκεται εντός των ετικετών *<BODY></BODY>*. Στο τμήμα αυτό, ορίζεται η θέση και τα χαρακτηριστικά των στοιχείων που περιλαμβάνει η ιστοσελίδα. Οι υπολογισμοί συντομότερης διαδρομής ξεκινούν μέσω του πλήκτρου *'Find'*, η επιλογή του οποίου καλεί τη συνάρτηση *calculate()* του Javascript κώδικα, εντός της οποίας καθορίζεται η διαδικασία διεξαγωγής των υπολογισμών.

Πριν τη δήλωση των βασικών στοιχείων που συνθέτουν το Javascript κώδικα (συναρτήσεις, μέθοδοι, κλάσεις κ.λπ.), δηλώνονται οι *global* μεταβλητές, οι οποίες είναι ορατές και μπορούν να χρησιμοποιηθούν από το σύνολο των συναρτήσεων που περιλαμβάνει ο Javascript κώδικας.

Η πρώτη συνάρτηση που ορίζεται στον κώδικα είναι η *readXML()*. Μέσω της συνάρτησης *readXML()*, διατυπώνεται στον *server* το αίτημα λήψης του XML αρχείου που περιέχει τα αποτελέσματα της αναζήτησης, με τη βοήθεια του αντικειμένου *XMLHttpRequest* και των μεθόδων *open()* και *send()* που καθορίζουν τον τύπο του αιτήματος και την αποστολή του στον *server* αντίστοιχα. Επιπλέον, δίνονται οι εντολές δημιουργίας του πίνακα στον οποίο πρόκειται να καταχωρηθεί η πληροφορία που περιέχει το XML έγγραφο. Οι λειτουργίες που υλοποιούνται στο XML έγγραφο και στον πίνακα καθορίζονται με τη βοήθεια των μεθόδων *getElementsByTagName()*, *getElementById()*, *removeChild()*, *createElement()*, και *appendChild()*.

Η επόμενη συνάρτηση είναι η *load()*, με τη βοήθεια τις οποίας ορίζονται η εισαγωγή του χάρτη στην ιστοσελίδα και οι λειτουργίες που λαμβάνουν χώρα στο χαρτογραφικό υπόβαθρο. Αρχικά, δημιουργείται ένα νέο αντικείμενο της βασικής κλάσης του Google Maps API V2, *GMap2*, μέσω της οποίας ορίζεται η δημιουργία ενός νέου χάρτη στην HTML ιστοσελίδα. Με τη βοήθεια της μεθόδου *addControl()*, καθορίζονται κάποιες βασικές λειτουργίες που αφορούν τις επιλογές για *zoom in* και *zoom out* στο χάρτη, την προσθήκη ελέγχου *overview* παραθύρου καθώς και τον τύπο του χάρτη (οδικός, δορυφορικός, υβριδικός) μέσα από την εισαγωγή ενός μενού στο επάνω δεξιό τμήμα του χάρτη. Το τμήμα κώδικα που ακολουθεί αφορά τη δημιουργία ενός *context menu*, με τη βοήθεια του οποίου εκτελείται μια σειρά λειτουργιών επάνω στο χάρτη. Ουσιαστικά, το συγκεκριμένο τμήμα του κώδικα επιτρέπει στο χρήστη την επιλογή των σημείων που ορίζουν την προς αναζήτηση διαδρομή. Η διαδικασία δημιουργίας του *context menu* ξεκινά με τη μέθοδο *createElement()* η οποία ορίζει τη δημιουργία του μενού, ενώ στη συνέχεια καθορίζονται οι ιδιότητές του και τα περιεχόμενά του. Το *context menu*, περιλαμβάνει τρεις επιλογές, τον ορισμό του σημείου αφετηρίας *(set origin)*, τον ορισμό του σημείου προορισμου *(set destination)* και το ορισμό ενός οποιουδήποτε άλλου σημείου επάνω στο χάρτη *(set point)*. Τα στοιχεία που περιλαμβάνει το *context menu* ορίζονται ως υπερσυνδέσεις *(ετικέτες <a href>)*. Παράλληλα, καθορίζονται ορισμένες ιδιότητες του *context menu*, τα όρια της περιοχής εντός της οποίας αυτό θα είναι ενεργό καθώς και τα συμβάντα που θα καθορίζουν την εμφάνιση και την απόκρυψή του.

Στο σημείο αυτό θα πρέπει να αναφερθεί ότι, η διαδικασία εισαγωγής των σημείων που επιλέγει ο χρήστης επάνω στο χάρτη μέσω του *context menu*, γίνεται με την κλήση αντίστοιχων συναρτήσεων οι οποίες καθορίζουν τα χαρακτηριστικά και τον τρόπο σήμανσης κάθε σημείου. Οι συναρτήσεις αυτές καλούνται από το τμήμα του κώδικα που ορίζει την εισαγωγή των σημείων που επιλέγει ο χρήστης στο χάρτη, μέσω του *context menu* που εμφανίζεται κάθε φορά που ο χρήστης επιλέγει με δεξί κλικ ένα σημείο στο χάρτη.

Η τελυταία βασική συνάρτηση που ορίζεται στον κώδικα είναι η *calculate()*, η οποία καλεί τη μέθοδο *getLatLng()*, που «ζητά» από τον *server* τον προσδιορισμό των συντεταγμένων των σημείων που έχουν επιλεγεί επάνω στο χάρτη καθώς και τις συναρτήσεις *get_field()*, *get_method()* και *get_ring()* προκειμένου να «αποσταλλούν» στον *server* οι παράμετροι βάσει των οποίων θα υλοποιηθούν οι απαραίτητοι

υπολογισμοί. Τέλος, η συνάρτηση *calculate()* καλεί και τη βιβλιοθήκη *jQuery* με τη βοήθεια της οποίας τα αποτελέσματα που προκύπτουν κάθε φορά στη βάση «μεταφέρονται» μέσω ενός XML και ενός KML αρχείου στην ιστοσελίδα, όπου και αναπαρίστανται ως λεκτική περιγραφή στο δυναμικό πίνακα όπου καταχωρούνται τα δεδομένα του XML εγγράφου και ως γεωμετρική οντότητα (*kml*) επάνω στο χαρτογραφικό υπόβαθρο.

Κώδικας εφαρμογής υλοποιημένος σε Javascript

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:v="urn:schemas-microsoft-com:vml">
  <head>
    <title>Test Map: Athens</title>
    <meta http-equiv="content-type" content="text/html;
charset=utf-8">


<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=t
rue_or_false&amp;key=ABQIAAAA0IRlHAPJIkACeS8zquQNvBT2yXp_ZAY8_
ufC3CFXhHIE1NvwkxSeC0aE3dEy-Om4TWI6tkQJoz4xlg"
type="text/javascript"></script>

<script type="text/javascript" src="jquery-1.6.1.js"></script>

<!-- java libraries for dragzoom -->
<script src="src/dragzoom.js" type="text/javascript"></script>

<!-- <script src="MStatusControl.js"></script> -->


<script type="text/javascript">



var origMarker;
var destMarker;
var pointMarkers = [];
var routeKML;
var map;
var field_name;
var method_name;
var roads_table;
var contextmenu;
var latlng;


//Opens an XML file and inserts it into the respective HTML
div
function readXML(xmlFile, tabDiv)
```

```
{
        if (window.XMLHttpRequest)
           xhttp=new XMLHttpRequest();
        else // Internet Explorer 5/6
           xhttp=new ActiveXObject("Microsoft.XMLHTTP");

        xhttp.open("GET",xmlFile+"?RANDOM=""+Math.random(),fals
e);
        xhttp.send("");
        xmlDoc=xhttp.responseXML;
        var xmlTable=xmlDoc.getElementsByTagName("Result");

        //Create a table structure that will be filled with
incoming values
        var res1 = document.getElementById(tabDiv);

        //Remove all attribute values from the respective
tabular spreadsheet
        while(res1.hasChildNodes())
        { res1.removeChild(res1.lastChild); }

        var tab = document.createElement('table');
        var tbo = document.createElement('tbody');
        var row, cell;

        //First table row is used to display attribute (field)
names
        row = document.createElement('tr');
        for(var j = 0; j < xmlTable[0].attributes.length ;j++)
        {     //Create a new cell for each attribute value
           cell = document.createElement('td');

        cell.appendChild(document.createTextNode(xmlTable[0].at
tributes[j].nodeName));
           row.appendChild(cell);
        }
        tbo.appendChild(row);


        //Iterate through all actual records and display their
contents
        for (var i = 0; i < xmlTable.length; i++)
        {
           row = document.createElement('tr');
           //Iterate through all columns of this record
           for(var j = 0; j < xmlTable[i].attributes.length
;j++)
           { //Create a new cell for each attribute value
                cell = document.createElement('td');

        cell.appendChild(document.createTextNode(xmlTable[i].ge
tAttribute(xmlTable[i].attributes[j].nodeName)));
                row.appendChild(cell);
           }
           tbo.appendChild(row);
```

```
        }

        tab.appendChild(tbo);
        tab.setAttribute("border", "2"); //Set cell border
        res1.appendChild(tab);
}


//Loads map and attaches basic event handlers
function load()
{

    if (GBrowserIsCompatible())
    {

        //Map setup
        map = new
GMap2(document.getElementById("map_canvas"));


//Code for zoom with frame
//First set of options is for the visual overlay (opacity: CSS
property for transparency)

        var boxStyleOpts = {
          opacity: .2,
          border: "2px solid red"
        }

//Second set of options is for everything else

        var otherOpts = {
          buttonHTML: "<img src='images/zoom-button.gif' />",
          buttonZoomingHTML: "<img src='images/zoom-button-
activated.gif' />",
          buttonStartingStyle: {width: '24px', height: '24px'}
        };


//Defining map controls

        map.addControl(new DragZoomControl(boxStyleOpts,
otherOpts));
        map.addControl(new GLargeMapControl3D());
        map.addControl(new GMapTypeControl());
        map.enableContinuousZoom();
        map.enableScrollWheelZoom();
        map.setCenter(new GLatLng(37.9765, 23.7292), 12);
        map.setMapType(G_HYBRID_MAP);

//Defining an Overview control widow

        map.addControl(new GOverviewMapControl(new
GSize(150,150)))
```

```
//Create a context menu

      contextmenu = document.createElement("div");
      contextmenu.style.visibility="hidden";
      contextmenu.style.background="#ffffff";
      contextmenu.style.border="1px solid #8888FF";

//Define actions for this context menu

contextmenu.innerHTML = '<a href="javascript:setOrigin()"><div
class="context">   Set origin
  <\/div><\/a><a
href="javascript:setDestination()"><div
class="context">   Set Destination
  <\/div><\/a><a href="javascript:setPoint()"><div
class="context">   Set point
  <\/div><\/a>';

         map.getContainer().appendChild(contextmenu);

//Listen for singleRightClick event


GEvent.addListener(map,"singlerightclick",function(pixel,tile)
{

//Store the "pixel" info in case we need it later
//Adjust the context menu location if near an edge
//Create a GControlPosition
//Apply it to the context menu, and make the context menu
visible

                clickedPixel = pixel;
                var x=pixel.x;
                var y=pixel.y;
                if (x > map.getSize().width - 120) { x =
map.getSize().width - 120 }
                if (y > map.getSize().height - 100) { y =
map.getSize().height - 100 }
                var pos = new
GControlPosition(G_ANCHOR_TOP_LEFT, new GSize(x,y));
                latlng =
map.fromContainerPixelToLatLng(pixel);
                pos.apply(contextmenu);
                contextmenu.style.visibility = "visible";
      });

//Close the context menu as soon as the user clicks somewhere
on the map

      GEvent.addListener(map, "click", function() {
                contextmenu.style.visibility="hidden";
      });
```

```
    }
}



function get_field()
{
for (var i=0; i < document.fieldform.field.length; i++)
   {
   if (document.fieldform.field[i].checked)
      field_name = document.fieldform.field[i].value;
   }
}


function get_method()
{
for (var j=0; j < document.methodform.method.length; j++)
   {
   if (document.methodform.method[j].checked)
      method_name = document.methodform.method[j].value;
   }
}

function get_ring()
{
   if (document.ringform.ring.checked)
        roads_table = "athens";
   else
        roads_table = "no_ring";

}

function setOrigin()
{
        if (latlng)
        {
                 if (origMarker)
                map.removeOverlay(origMarker);

           var greenIcon = new GIcon(G_DEFAULT_ICON);
           greenIcon.image =
"http://www.google.com/intl/en_us/mapfiles/ms/micons/green-
dot.png";
           origMarker = new GMarker(latlng, greenIcon);
           map.addOverlay(origMarker);
           contextmenu.style.visibility="hidden";

           GEvent.addListener(origMarker, "click", function()
           {
               map.removeOverlay(origMarker);
                    origMarker=null;
           });
```

```
        }
}


function setDestination()
{
        if (latlng)
        {
                if (destMarker)
               map.removeOverlay(destMarker);

           var redIcon = new GIcon(G_DEFAULT_ICON);
           redIcon.image =
"http://www.google.com/intl/en_us/mapfiles/ms/micons/red-
dot.png";
           destMarker = new GMarker(latlng, redIcon);
           map.addOverlay(destMarker);
           contextmenu.style.visibility="hidden";

           GEvent.addListener(destMarker, "click", function()
           {
               map.removeOverlay(destMarker);
                    destMarker=null;
           });
        }
}

function setPoint()
{
        if (latlng)
        {
           var blueIcon = new GIcon(G_DEFAULT_ICON);
           blueIcon.image =
"http://www.google.com/intl/en_us/mapfiles/ms/micons/blue-
dot.png";
           var pointMarker = new GMarker(latlng, blueIcon);
           map.addOverlay(pointMarker);
           //Store the reference
           pointMarkers.push(pointMarker);
           contextmenu.style.visibility="hidden";

           GEvent.addListener(pointMarker, "click", function()
           {
                //Remove that marker only
                for (var i=0; i<pointMarkers.length; i++)
                {
                    if
(pointMarkers[i].getLatLng().equals(pointMarker.getLatLng()))
                    {
                          map.removeOverlay(pointMarkers[i]);
                               pointMarkers.splice(i,1);
//alert("removed!");
                               break;
                          }
                }
```

```
            });

        }


}

//Sends a request to the server according to input coordinates


function calculate()
{
        var origLatLng;
        var destLatLng;
        var pointLatLng;
        var result = "?";

        get_field();
        get_method();
        get_ring();


        if (method_name == "tsp")
        {
           if (pointMarkers.length>0)
           {
                if (origMarker)
                {
                     origLatLng = origMarker.getLatLng();
                     result = "?coords=" + origLatLng.lng() +
"|" + origLatLng.lat();
                }
                else
                {
                     alert("No origin specified!");
                     return;
                }

                for (var i=0; i<pointMarkers.length; i++)
                {
                     pointLatLng =
pointMarkers[i].getLatLng();
                     result = result + "|" + pointLatLng.lng()
+ "|" + pointLatLng.lat();
                }

           }

           result = result + "&roads=" + roads_table +
"&delta=0.1" + "&method=" + method_name + "&field=" +
field_name;

           //NOTE: use the jQuery library in order to fetch
results from the database
```

```
            jQuery.get( "routingtsp.php" +  result
+"&RANDOM=""+Math.random(), function ( data )
                  {
                  if (routeKML)
                        map.removeOverlay(routeKML);
                  //display results into the respective table

         readXML('http://localhost/output.xml','table1');
                  routeKML = new
GGeoXml("http://147.102.106.224/route.kml"+"?RANDOM=""+Math.ra
ndom());
                  map.addOverlay(routeKML);

                  });
         }
         else
         {

            if (origMarker)
            {
                  origLatLng = origMarker.getLatLng();
                  result = result + "x1=" + origLatLng.lng() +
"&y1=" + origLatLng.lat();
            }
            else
                  result = result + "x1=0&y1=0";


            if (destMarker)
            {
                  destLatLng = destMarker.getLatLng();
                  result = result +"&x2=" + destLatLng.lng() +
"&y2=" + destLatLng.lat();
            }
            else
                  result = result + "&x2=0&y2=0";


            result = result + "&roads=" + roads_table +
"&delta=0.0" + "&method=" + method_name + "&field=" +
field_name;

            //NOTE: use the jQuery library in order to fetch
results from the database
            jQuery.get( "routing.php" +  result
+"&RANDOM=""+Math.random(), function ( data )
                  {
                  if (routeKML)
                        map.removeOverlay(routeKML);
                  //display results into the respective table

         readXML('http://localhost/output.xml','table1');
                  routeKML = new
GGeoXml("http://147.102.106.224/route.kml"+"?RANDOM=""+Math.ra
ndom());
```

```
                map.addOverlay(routeKML);

                });
        }

}

</script>

<META content="MSHTML 6.00.2800.1613" name=GENERATOR></head>

<BODY onload="load()" onunload="GUnload()">

<TABLE>
<TBODY>
<Font face="Book Antiqua, Arial">
<Body bgcolor="  #F0F7D2"
<table align="left" width="20%" bgcolor="white"
cellspacing="4" cellpadding="2" border="2">
<TR><TD>
        <Font face="Book Antiqua, Arial" color="CadetBlue">
      <H2> Routing with GoogleMaps </H2>

<BR>
</TD>
</TR>
</table>

<table align="left" width=100% cellspacing="6"
cellpadding="5">
<TR><TD align=left valign="top">
<div id="map_canvas" style="WIDTH: 720px; HEIGHT:
420px"></div><BR>
<table align="left" width=60% cellspacing="6" cellpadding="1">
<TR><TD valign="top">
<INPUT TYPE=BUTTON OnClick="calculate();" VALUE="Find"></TD>
<TD valign="top"><form name="methodform">
<input type="radio" name="method" value="dijkstra"
checked/>Dijkstra Algorithm <br/>
<input type="radio" name="method" value="astar" /> Astar
Algorithm<br/>
<input type="radio" name="method" value="tsp" /> TSP
Algorithm<br/>
</form> </TD>

<TD valign="top"><form name="fieldform">
<input type="radio" name="field" value="cost" checked/>Time
<br/>
<input type="radio" name="field" value="length" /> Length
<br/>
</form> </TD>


<TD valign="top"><form name="ringform">
```

```
<input type="checkbox" name="ring" value="traffic" checked
/>Use Roads within Ring<br />
</form>
<BR>
</TD>
</TR>
</table>
</TD>
<TD valign="top">
<font size="2">
<div id="table1"></div>
</font>
<BR>
</TD>
</TR>
</table>


</BODY>
</html>
```

IV. Κώδικες σύνδεσης της ιστοσελίδας με τη βάση δεδομένων

Στην ενότητα αυτή παρουσιάζονται οι δύο PHP κώδικες μέσω των οποίων επιτυγχάνεται η σύνδεση της ιστοσελίδας με τη βάση δεδομένων. Πρόκειται για τους κώδικες *'routing.php'* και *'routingtsp.php'*. Ο πρώτος κώδικας μεταβιβάζει στη βάση ένα αίτημα για τον υπολογισμό της βέλτιστης διαδρομής με τη βοήθεια των αλγόριθμων Dijkstra και A*, ενώ ο δεύτερος μεταβιβάζει ένα αίτημα υπολογισμού συντομότερης διαδρομής με τη βοήθεια του αλγόριθμου Traveling Sales Person. Η σύνδεση με τη βάση για την εκκίνηση των υπολογισμών επιτυγχάνεται μέσω των ανάλογων php συναρτήσεων, ενώ τα αποτελέσματα που προκύπτουν καταχωρούνται σε ένα XML και ένα KML αρχείο. Η δημιουργία των δύο αυτών αρχείων υλοποιείται μέσα από τον ορισμό στιγμιοτύπων της κλάσης *DOMDocument()*. Στην πρώτη περίπτωση, το στιγμιότυπο που δημιουργείται καταχωρείται στη μεταβλητή *$dom* και χρησιμοποιείται για την ανάλυση του XML αρχείου. Στη δεύτερη περίπτωση, το στιγμιότυπο που δημιουργείται, καταχωρείται στη μεταβλητή *$kml* και στη συνέχεια ακολουθούν οι μέθοδοι βάσει των οποίων υλοποιείται η δημιουργία του KML αρχείου και η καταχώρηση των γεωμετρικών στοιχείων σε αυτό.

Κώδικας 'routing.php'

```
<?php header ('Content-type: application/vnd.google-
earth.kml+xml; charset=UTF-8');
$dom = new DOMDocument("1.0");
$node = $dom->createElement("output");
$parnode = $dom->appendChild($node);

// Start KML file, create parent node
$kml = new DOMDocument('1.0','UTF-8');

//Create the root KML element and append it to the Document
$kmlnode = $kml-
>createElementNS('http://earth.google.com/kml/2.2','kml');
$pNode = $kml->appendChild($kmlnode);

$dNode = $kml->createElement('Document');
$docNode = $pNode->appendChild($dNode);


//Create a Folder element and append it to the KML element
$fnode = $kml->createElement('Folder');
$folderNode = $docNode->appendChild($fnode);

$source_x=$_GET["x1"];
```

```
$source_y=$_GET["y1"];
$target_x=$_GET["x2"];
$target_y=$_GET["y2"];
$method=$_GET["method"];
$delta=$_GET["delta"];
$field=$_GET["field"];
$roads=$_GET["roads"];

if ($delta >0)
   $delta = "," . $delta . ", 'cost'";
else
   $delta = "";

$db = pg_connect('host=localhost dbname=testgis user=postgres
password=postgres') or die('Could not connect: ' .
pg_last_error());

function getNode($x1, $y1)
{
        $query = "SELECT node_id, x, y, sqrt((x-" . $x1 . ")*
(x-" . $x1 . ") + (y-" . $y1 . ")*(y-" . $y1 . ")) AS dist
FROM athens_nodes WHERE x>(" . $x1 . "-0.2) AND x<(" . $x1 .
"+0.2) AND y>(" . $y1 . "-0.2) AND y<(" . $y1 . "+0.2) ORDER
BY dist ASC LIMIT 1";

        $result = pg_query($query) or die('Query failed: ' .
pg_last_error());

        while ($row = pg_fetch_array($result, null,
PGSQL_ASSOC))
        {
           $node = $row['node_id'];
        }

        // Free resultset
        pg_free_result($result);

    return $node;
}


$source= getNode($source_x, $source_y);
$target= getNode($target_x, $target_y);


        $query = "SELECT id, gid, cost, greekname, latinname,
ST_AsKML(the_geom) AS shape FROM " . $method .
"_sp_directed_cost('" . $roads . "'," . $source . ", " .
$target . $delta . ", true, true, '" . $field .
"','resNodes')";
//echo $query;
        $output = pg_query($query) or die('Query failed: ' .
pg_last_error());
```

```
        while ($row = pg_fetch_array($output, null,
PGSQL_ASSOC))
{

        // ADD TO XML DOCUMENT NODE
        $node = $dom->createElement("Result");
        $newnode = $parnode->appendChild($node);
        $newnode->setAttribute("id", $row['id']);
        $newnode->setAttribute("greekname", $row['greekname']);
        $newnode->setAttribute("latinname", $row['latinname']);
        $newnode->setAttribute("cost", $row['cost']);

        $shape = $row['shape'];

        // ADD TO KML DOCUMENT NODE
        //Create a Placemark and append it to the document
        $kmlnode = $kml->createElement('Placemark',$shape);
        $placeNode = $folderNode->appendChild($kmlnode);

}

        //Save the entire XML file
        $dom->save('output.xml');

        //Save the entire KML file
        $kmlOutput = $kml->saveXML();
        $kmlOutput = str_replace("&lt;" , "<" , $kmlOutput);
        $kmlOutput = str_replace("&gt;" , ">" , $kmlOutput);
        $fp = fopen('route.kml', 'w');
        fwrite($fp, $kmlOutput);
        fclose($fp);

        // Free outputtset
        pg_free_result($output);

        // Closing connection
        pg_close($db);
?>
```

Κώδικας 'routingtsp.php'

```
<?php header ('Content-type: application/vnd.google-
earth.kml+xml; charset=UTF-8');
$dom = new DOMDocument("1.0");
$node = $dom->createElement("output");
$parnode = $dom->appendChild($node);

// Start KML file, create parent node
$kml = new DOMDocument('1.0','UTF-8');

//Create the root KML element and append it to the Document
```

```
$kmlnode = $kml-
>createElementNS('http://earth.google.com/kml/2.2','kml');
$pNode = $kml->appendChild($kmlnode);

$dNode = $kml->createElement('Document');
$docNode = $pNode->appendChild($dNode);


//Create a Folder element and append it to the KML element
$fnode = $kml->createElement('Folder');
$folderNode = $docNode->appendChild($fnode);

$coords_str = $_GET["coords"];
$coords = explode("|",$coords_str);

$method=$_GET["method"];
$delta=$_GET["delta"];
$field=$_GET["field"];
$roads=$_GET["roads"];

$db = pg_connect('host=localhost dbname=testgis user=postgres
password=postgres') or die('Could not connect: ' .
pg_last_error());

function getNode($x1, $y1)
{
        $query = "SELECT node_id, x, y, sqrt((x-" . $x1 . ")*
(x-" . $x1 . ") + (y-" . $y1 . ")*(y-" . $y1 . ")) AS dist
FROM athens_nodes WHERE x>(" . $x1 . "-0.2) AND x<(" . $x1 .
"+0.2) AND y>(" . $y1 . "-0.2) AND y<(" . $y1 . "+0.2) ORDER
BY dist ASC LIMIT 1";

        $result = pg_query($query) or die('Query failed: ' .
pg_last_error());

        while ($row = pg_fetch_array($result, null,
PGSQL_ASSOC))
        {
           $node = $row['node_id'];
        }

        // Free resultset
        pg_free_result($result);

    return $node;
}


$ids="";
for($i=0;$i<count($coords);$i=$i+2)
{
        $point = getNode($coords[$i], $coords[$i+1]);

        if ($i==0)
           $source = intval($point);
```

```
        $ids= $ids . ',' . $point ;
}

$ids = substr($ids, 1);


        $query = "SELECT id, gid, cost, greekname, latinname,
ST_AsKML(the_geom) AS shape FROM " . $method .
"_sp_directed_cost('" . $roads . "','" . $ids . "', " .
$source . "," . $delta . ", true, true, '" . $field .
"','resNodes')";
echo $query;
        $output = pg_query($query) or die('Query failed: ' .
pg_last_error());

        while ($row = pg_fetch_array($output, null,
PGSQL_ASSOC))
{

        // ADD TO XML DOCUMENT NODE
        $node = $dom->createElement("Result");
        $newnode = $parnode->appendChild($node);
        $newnode->setAttribute("id", $row['id']);
        $newnode->setAttribute("greekname", $row['greekname']);
        $newnode->setAttribute("latinname", $row['latinname']);
        $newnode->setAttribute("cost", $row['cost']);

        $shape = $row['shape'];

        // ADD TO KML DOCUMENT NODE
        //Create a Placemark and append it to the document
        $kmlnode = $kml->createElement('Placemark',$shape);
        $placeNode = $folderNode->appendChild($kmlnode);

}

        //Save the entire XML file
        $dom->save('output.xml');

        //Save the entire KML file
        $kmlOutput = $kml->saveXML();
        $kmlOutput = str_replace("&lt;" , "<" , $kmlOutput);
        $kmlOutput = str_replace("&gt;" , ">" , $kmlOutput);
        $fp = fopen('route.kml', 'w');
        fwrite($fp, $kmlOutput);
        fclose($fp);

        // Free outputtset
        pg_free_result($output);

        // Closing connection
        pg_close($db);
?>
```